

ROBUSTNESS AND OPTIMIZATION OF SCRIP SYSTEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ian Alexander Kash

February 2010

© 2010 Ian Alexander Kash
ALL RIGHTS RESERVED

ROBUSTNESS AND OPTIMIZATION OF SCRIP SYSTEMS

Ian Alexander Kash, Ph.D.

Cornell University 2010

A game theoretic model of scrip systems is analyzed. It is shown that scrip systems have pure strategy equilibria in a natural class of strategies. An algorithm is given that can compute such an equilibrium and the resulting distribution of money (scrip). The effect of varying the total amount of money in the system on efficiency (i.e., social welfare—the total utility of all the agents in the system) is analyzed, and it is shown that by maintaining the appropriate ratio between the total amount of money and the number of agents, efficiency is maximized. This ratio can be found by increasing the money supply up to the point that the system experiences a “monetary crash,” where money is sufficiently devalued that no agent is willing to perform a service. The implications of the presence of altruists, hoarders, sybils, and collusion on the performance of the system are examined. In practice, agents in a scrip system will not have the necessary information to compute this equilibrium. However, a simple learning algorithm is investigated and simulation results are presented that show it enables agents to converge to equilibrium.

BIOGRAPHICAL SKETCH

Ian Alexander Kash was born September 14, 1982. After graduating from Pleasantville High School, he attended Carnegie Mellon University. In May 2004, he received the Bachelor of Science degree in Computer Science with additional majors in Mathematics and Philosophy.

To my parents.

ACKNOWLEDGEMENTS

I would like to thank Randy Farmer, Peter Harremoes, Shane Henderson, Jon Kleinberg, David Parkes, Dan Reeves, Michael Wellman, and anonymous referees for helpful suggestions, discussions, and criticisms. Emin Gün Sirer and Éva Tardos have served on my committee and provided me with valuable feedback. My advisors, Eric Friedman and Joseph Halpern, have been a tremendous source of ideas, support, and guidance, and have patiently read the many drafts of this work in all of its previous forms. Finally, I would like to thank my family and friends for all their encouragement and support. This research was supported in part by NSF under grants ITR-0325453, CDI-0835706, CTC-0208535, IIS-0812045, and IIS-0534064; by ONR under grant N00014-01-10-511; by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grants N00014-01-1-0795 and N00014-04-1-0725; and by AFOSR under grants F49620-02-1-0101, FA9550-08-1-0438, FA9550-05-1-0055, and FA9550-09-1-0266.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	viii
1 Introduction and Related Work	1
1.1 Introduction	1
1.2 Related Work	9
2 A Model of a Scrip System	17
2.1 The Model	17
2.2 Analyzing the Distribution of Wealth	24
2.3 Existence of Equilibria	28
2.4 Simulations	35
3 Implications of the Model	38
3.1 Social Welfare and Scalability	38
3.2 Dealing with Nonstandard Agents	45
3.2.1 Altruists	45
3.2.2 Hoarders	50
3.2.3 Sybils	53
3.2.4 Collusion	60
3.3 Identifying User Strategies	64
4 Learning in Scrip Systems	71
4.1 Introduction to Learning in Distributed Systems	71
4.2 Theoretical Results	75
4.2.1 Large Anonymous Games	75
4.2.2 Best-Reply Dynamics	77
4.2.3 Stage Learners	78
4.2.4 Convergence Theorem	81
4.3 Simulation Results	84
4.3.1 A Contribution Game	86
4.3.2 A Congestion Game	90
4.3.3 Scrip Systems	91
5 Conclusion	94
A Appendix	103
A.1 Proof of Theorem 2.2.1	103
A.2 Proofs from Section 2.3	111
A.3 Proofs from Section 3.2	124

A.4 Proof of Lemma 3.3.2	125
Bibliography	127

LIST OF FIGURES

2.1	A hypothetical best-reply function with one type of agent.	33
2.2	Maximum distance from minimum relative entropy distribution over 10^6 timesteps.	35
2.3	Distance from minimum relative entropy distribution with 1000 agents.	36
2.4	Average time to get within .001 of the minimum relative entropy distribution.	37
3.1	Social welfare for various average amounts of money, demon- strating a monetary crash.	44
3.2	Altruists can cause a crash.	51
3.3	m can be adjusted as a increases.	51
3.4	The effect of p_u on utility	55
3.5	The effect of sybils on utility	55
3.6	Sybils can improve utility	56
3.7	Sybils can cause a crash	57
3.8	The effect of collusion on utility	62
3.9	Distribution of money with two types of agents.	69
3.10	Log of the distribution of money with two types of agents.	69
4.1	Stage Learners with Random Matching	87
4.2	Hart and Mas-Colell with Random Matching	87
4.3	Exp3 with Random matching	88
4.4	Stage Learning with Average-Based Payoffs	89
4.5	Stage Learners in a Congestion Game	90
4.6	Stage Learners in a Scrip System	92
4.7	A Scrip System with Churn	92
4.8	A Scrip System with Different Stage Lengths	93

CHAPTER 1

INTRODUCTION AND RELATED WORK

1.1 Introduction

Historically, non-governmental organizations have issued their own currencies for a wide variety of purposes. These currencies, known as *scrip*, have been used in company towns where government issued currency was scarce [72], in Washington DC to reduce the robbery rate of bus drivers [77], and in Ithaca (New York) to promote fairer pay and improve the local economy [45]. Scrip systems are also becoming more prevalent in online systems.

To give some examples, market-based solutions using scrip systems have been suggested for applications such as system-resource allocation [58], managing replication and query optimization in a distributed database [68], and allocating experimental time on a wireless sensor network test bed [17]; a number of sophisticated scrip systems have been proposed [35, 44, 75] to allow agents to pool resources while avoiding what is known as *free riding*, where agents take advantage of the resources the system provides while providing none of their own (as Adar and Huberman [2] have shown, this behavior certainly takes place in systems such as Gnutella); and Yootles [63] uses a scrip system as a way of helping groups make decisions using economic mechanisms without involving real money.

Creating a scrip system creates a new market where goods and services can be exchanged that may have been impractical or undesirable to implement with standard currency. However, the potential benefits of a scrip system will not

necessarily be realized simply by creating the framework to support one. The story of the Capitol Hill Baby Sitting Co-op [69], popularized by Krugman [50], provides a cautionary tale. The Capitol Hill Baby Sitting Co-op was a group of parents working on Capitol Hill who agreed to cooperate to provide babysitting services to each other. In order to enforce fairness, they issued a supply of scrip with each coupon worth a half hour of babysitting. At one point, the co-op had a recession. Many people wanted to save up coupons for when they wanted to spend an evening out. As a result, they went out less and looked for more opportunities to babysit. Since a couple could earn coupons only when another couple went out, no one could accumulate more, and the problem only got worse.

After a number of failed attempts to solve the problem, such as mandating a certain frequency of going out, the co-op started issuing more coupons. The results were striking. Since couples had a sufficient reserve of coupons, they were more comfortable spending them. This in turn made it much easier to earn coupons when a couple's supply got low. Unfortunately, the amount of scrip grew to the point that most of the couples felt "rich." They had enough scrip for the foreseeable future, so naturally they didn't want to devote their evening to babysitting. As a result, couples who wanted to go out were unable to find another couple willing to babysit.

This anecdote shows that the amount of scrip in circulation can have a significant impact on the effectiveness of a scrip system. If there is too little money in the system, few agents will be able to afford service. At the other extreme, if there is too much money in the system, people feel rich and stop looking for work. Both of these extremes lead to inefficient outcomes. This suggests that

there is an optimal amount of money, and that nontrivial deviations from the optimum towards either extreme can lead to significant degradation in the performance of the system.

In this paper, we provide a formal model in which to analyze scrip systems. We describe a simple scrip system and show that, under reasonable assumptions, for each fixed amount of money there is a nontrivial equilibrium involving *threshold strategies*, where an agent accepts a request if he has less than $\$k$ for some threshold k .¹

An interesting aspect of our analysis is that, in equilibrium, the distribution of users with each amount of money is the distribution that minimizes relative entropy to an appropriate distribution (subject to the money supply constraint). This allows us to use techniques from statistical mechanics to explicitly compute the distribution of money and thus agents' best-reply functions.

An understanding of agents' best-reply functions allows us to compute the money supply that maximizes social welfare, given the number of agents. As we show, adding more money decreases the equilibrium number of agents with no money, thus increasing social welfare. However, this only works up to a point. Once a critical amount of money is reached, the system experiences a monetary crash: there is so much money that, in equilibrium, everyone will feel rich and no agents are willing to work. We show that, to get optimal performance, we want the total amount of money in the system to be as close as possible to the critical amount, but not to go over it. If the amount of money in the system is over the critical amount, we get the worst possible equilibrium, where all agents have utility 0. This provides a significant tradeoff between efficiency

¹Although we refer to our unit of scrip as the dollar, these are not real dollars nor do we view them as convertible to dollars.

and robustness.

Our equilibrium analysis assumes that all agents have somewhat similar motivation: in particular, they do not get pleasure simply from performing a service, and are interested in money only to the extent that they can use it to get services performed. But in real systems, not all agents have this motivation. Some of the more common “nonstandard” agents are *altruists* and *hoarders*. Altruists are willing to satisfy all requests, even if they go unpaid; hoarders value scrip for its own sake and are willing to accumulate amounts far beyond what is actually useful. Studies of the Gnutella peer-to-peer file-sharing network have shown that one percent of agents satisfy fifty percent of the requests [2, 43]. These agents are doing significantly more work for others than they will ever have done for them, so can be viewed as altruists. Anecdotal evidence also suggests that the introduction of any sort of currency seems to inspire hoarding behavior on the part of some agents, regardless of the benefit of possessing money. For example, SETI@home has found that contributors put in significant effort to make it to the top of their contributor rankings. This has included returning fake results of computations rather than performing them [82].

Altruists and hoarders have opposite effects on a system: having altruists has the same effect as adding money; having hoarders is essentially equivalent to removing it. With enough altruists in the system, the system has a monetary crash, in the sense that standard agents stop being willing to provide service, just as when there is too much money in the system. We show that, until we get to the point where the system crashes, the utility of standard agents is improved by the presence of altruists. We show that the presence of altruists makes the critical point lower than it would without them. Thus, a system designer trying

to optimize the performance of the system by making the money supply as close as possible to the critical point (but under it, since being over it would result in a “crash”) needs to be careful about estimating the number of altruists.

Similarly, we show that, as the fraction of hoarders increases, standard agents begin to suffer because there is effectively less money in circulation. On the other hand, hoarders can improve the stability of a system. Since hoarders are willing to accept an infinite amount of money, they can prevent the monetary crash that would otherwise occur as money was added. In any case, our results show how the presence of both altruists and hoarders can be mitigated by appropriately controlling the money supply.

Beyond nonstandard agents, we also consider two different manipulative behaviors in which standard agents may engage: creating multiple identities, known as *sybils* [22], and collusion. In scrip systems where each new user is given an initial amount of scrip, there is an obvious benefit to creating sybils. Even if this incentive is removed, sybils are still useful: they can be used to increase the likelihood that an agent will be asked to provide service, which makes it easier for him to earn money. This increases the utility of the sybiling agent, at the expense of other agents. From the perspective of an agent considering creating sybils, the first few sybils can provide him with a significant benefit, but the benefits of additional sybils rapidly diminish. So if a designer can make sybiling moderately costly, the number of sybils actually created by rational agents will usually be relatively small.

If a small fraction of agents have sybils, the situation is more subtle. Agents with sybils still do better than those without, but the situation is not zero-sum. In particular, even agents without sybils might be better off, due to having more

opportunities to earn money. Somewhat surprisingly, sybils can actually result in everyone being better off. However, exploiting this fact is generally not desirable. The same process that leads to an improvement in social welfare can also lead to a monetary crash, where all agents stop providing service. The system designer can achieve the same effects by increasing the average amount of money or biasing the volunteer selection process. In practice, it seems better to do this than to exploit the possibility of sybils.

In our setting, having sybils is helpful because it increases the likelihood that an agent will be asked to provide service. Our analysis of sybils applies no matter how this increase in likelihood occurs. In particular, it applies to advertising. Thus, our results suggest that there are tradeoffs involved in allowing advertising. For example, many systems allow agents to announce their connection speed and other similar factors. If this biases requests towards agents with high connection speeds, even when agents with lower connection speeds are perfectly capable of satisfying a particular request, then agents with low connection speeds will have an unnecessarily worsened experience in the system. This also means that such agents will have a strong incentive to lie about their connection speed.

While collusion is generally a bad thing, in the context of scrip systems with fixed prices, it is almost entirely positive. Without collusion, if a user runs out of money he is unable to request service until he is able to earn some. However, a colluding group can pool their money so that all members can make a request whenever the group as a whole has some money. This increases welfare for the agents who collude because agents who have no money receive no service. Collusion tends to benefit the non-colluding agents as well. Since colluding

agents work less often, it is easier for everyone to earn money, which ends up making everyone better off. However, as with sybils, collusion does have the potential of crashing the system if the average amount of money is high.

While a designer should generally encourage collusion, we would expect that in most systems there will be relatively little collusion and what collusion exists will involve small numbers of agents. After all, scrip systems exist to try and resolve resource-allocation problems where agents are competing with each other. If they could collude to optimally allocate resources within the group, they would not need a scrip system in the first place. However, many of the benefits of collusion come from agents being allowed to effectively have a negative amount of money (by borrowing from their collusive partners). These benefits could also be realized if agents are allowed to borrow money, so designing a loan mechanism could be an important improvement for a scrip system. Of course, implementing such a loan mechanism in a way that prevents abuse requires a careful design.

In order to effectively utilize our results, a system designer needs to be able to identify characteristics of agents (with what frequency do they make requests, how likely are they to be chosen to satisfy a request, and so on) and what strategies they are following. This is particularly useful because finding an amount of money close to the point of monetary crash, but not past it, relies on an understanding of the agents in the system. Of course, such information is also of great interest to social scientists and marketers. We show how relatively simple observations of the system can be used to infer this information.

It is not only the system designer that does not know the characteristics of the agents in the system; agents will generally be unaware of each other's

characteristics. This raises a natural question: how will agents find the equilibrium strategies in a scrip system? In Chapter 4, we investigate the more general problem of agent's learning what strategy to use in a distributed system. While multiagent learning lacks practical general algorithms, distributed systems have several simplifying characteristics. First, there are a large number of agents learning. Second, they are *anonymous*, agents' payoffs depend on the number of (each type of) agent that plays each strategy rather than on exactly which agents play which strategy. Third, in many systems, and in particular in scrip systems, best-reply dynamics converge. We show that, in such games, simple algorithms suffice to allow agents to learn equilibria.

We prove convergence results for an algorithm known as *stage learning* [27]. An agent using the stage learning algorithm divides the rounds of the game into a series of stages. In each stage, he uses a fixed strategy except that he occasionally explores. At the end of a stage, he chooses as his strategy for the next stage whatever strategy had the highest average reward in the current stage.

To go with these theoretical results, we provide simulations to show that stage learning is effective in practice. While the theoretical results about stage learning are restricted to simpler games and do not directly apply to scrip systems, these simulations show that stage learning still allows agents to converge to equilibrium. Our results also demonstrate that stage learning is robust to factors such as churn (agents joining and leaving the system) and asynchrony (agents using stages of different lengths).

1.2 Related Work

Scrip systems have a long history in computer science, with two main thrusts: resource allocation and free-riding prevention. Early applications for resource allocation include agoric systems [58], which envisioned solving problems such as processor scheduling using markets, and Mariposa [68], a market-driven query optimizer for distributed databases. More recently, scrip systems have been used to allocate the resources of research testbeds. Examples include Mirage [17] for wireless sensor networks, Bellagio [7] for PlanetLab, and Egg [15] for grid computing. Virtual markets have been used to coordinate the activity of nodes of a sensor network [52]. Yootles [63] uses a scrip to help people make everyday decisions, such as where to have lunch, without involving real money.

Systems that use scrip to prevent free riding include KARMA [75], which provides a general framework for P2P networks. Gupta et al. [35] propose what they call a “debit-credit reputation computation” for P2P networks, which is essentially a scrip system. Fileteller [44] uses payments in a network file storage system. Dandelion [67] uses scrip in a content distribution setting. Belenkiy et al. [8] consider how a BitTorrent-like system can make use of e-cash. Antfarm [61] uses scrip to optimize content distribution across a number of BitTorrent-like swarms.

Despite this tremendous interest in scrip systems, there has been relatively little work studying how they behave. Chun et al. [59] studied user behavior in a deployed scrip system and observed that users tried various (rational) manipulations of the auction mechanism used by the system. Their observations suggest that system designers will have to deal with game-theoretic concerns.

Hens et al. [40] do a theoretical analysis of what can be viewed as a scrip system in a related model. There are a number of significant differences between the models. First, in the Hens et al. model, there is essentially only one type of agent, but an agent's utility for getting service (our γ_i) may change over time. Thus, at any time, there will be agents who differ in their utility. At each round, we assume that one agent is chosen (by nature) to make a request for service, while other agents decide whether or not to provide it. In the Hens et al. model, at each round, each agent decides whether to provide service, request service, or opt out, as a function of his utilities and the amount of money he has. They assume that there is no cost for providing service and everyone is able to provide service. Under this assumption, a system with optimal performance is one where half the agents request service and the other half are willing to provide it. Despite these differences, Hens et al. also show that agents will use a threshold strategy. However, although they have inefficient equilibria, because there is no cost for providing service, their model does not exhibit the monetary crashes that our model can exhibit.

Aperjis and Johari [4] examine a model of a P2P filesharing system as an exchange economy. They associate a price (in bandwidth) with each file and find a market equilibrium in the resulting economy. However, they do not use an explicit currency.

The effects of altruists, sybils, and collusion on system behavior have all been studied in other contexts. Work on the evolution of cooperation stresses the importance of altruists willing to undertake costly punishment [38]. Yokoo et al. [80] studied the effects of sybils in auctions. Solution concepts such as *strong Nash equilibrium* [6] and *k-t robust equilibrium* [1] explicitly address collusion in

games; Hayrapetyan et al. [39] study collusion in congestion games and find cases where, as with scrip systems, collusion is actually beneficial.

The ultimate goal of a scrip systems is to promote cooperation. While there is limited theoretical work on scrip systems, there is a large body of work on cooperation. Much of the work involves a large group of agents being randomly matched to play a game such as prisoner's dilemma. Such models were studied in the economics literature [49, 23] and first applied to online reputations in [26]; Feldman et al. [24] apply them to P2P systems.

These models fail to capture important asymmetries in the interactions of the agents. When a request is made, there are typically many people in the network who can potentially satisfy it (especially in a large P2P network), but not all can. For example, some people may not have the time or resources to satisfy the request. The random-matching process ignores the fact that some people may not be able to satisfy the request. (Presumably, if the person matched with the requester could not satisfy the match, he would have to defect.) Moreover, it does not capture the fact that the decision as to whether to "volunteer" to satisfy the request should be made before the matching process, not after. That is, the matching process does not capture the fact that if someone is unwilling to satisfy the request, there will doubtless be others who can satisfy it. Finally, the actions and payoffs in prisoner's dilemma game do not obviously correspond to actual choices that can be made. For example, it is not clear what defection on the part of the requester means. Our model addresses all these issues.

Scrip systems are not the only approach to preventing free riding. Two other approaches often used in P2P networks are barter and reputation systems. Perhaps the best-known example of a system that uses barter is BitTorrent [19],

where clients downloading a file try to find other clients with parts they are missing so that they can trade, thus creating a roughly equal amount of work. Since the barter is restricted to users currently interested in a single file, this works well for popular files, but tends to have problems maintaining availability of less popular ones. An example of a barter-like system built on top of a more traditional file-sharing system is the credit system used by eMule. Each user tracks his history of interactions with other users and gives priority to those he has downloaded from in the past. However, in a large system, the probability that a pair of randomly-chosen users will have interacted before is quite small, so this interaction history will not be terribly helpful. Anagnostakis and Greenwald [3] present a more sophisticated version of this approach, but it still seems to suffer from similar problems.

A number of attempts have been made at providing general reputation systems (e.g. [34, 35, 48, 79]). The basic idea is to aggregate each user's experience into a global number for each individual that intuitively represents the system's view of that individual's reputation. However, these attempts tend to suffer from practical problems because they implicitly view users as either "good" or "bad", assume that the "good" users will act according to the specified protocol, and that there are relatively few "bad" users. Unfortunately, if there are easy ways to game the system, once this information becomes widely available, rational users are likely to make use of it. We cannot count on only a few users being "bad" (in the sense of not following the prescribed protocol). For example, Kazaa uses a measure of the ratio of the number of uploads to the number of downloads to identify good and bad users. However, to avoid penalizing new users, they gave new users an average rating. Users discovered that they could use this relatively good rating to free ride for a while and, once it started

to get bad, they could delete their stored information and effectively come back as a “new” user, thus circumventing the system (see [3] for a discussion and [26] for a formal analysis of this “whitewashing”). Thus, Kazaa’s reputation system is ineffective.

Chapter 4 explores how agents can learn equilibrium strategies in a distributed system such as a scrip system. One approach to learning to play games is to generalize reinforcement learning algorithms such as Q-learning [78]. One nice feature of this approach is that it can handle games with state, which is important in distributed systems. In Q-learning, an agent associates a value with each state-action pair. When he chooses action a in state s , he updates the value $Q(s, a)$ based on the reward he received and the best value he can achieve in the resulting state s' ($\max_{a'} Q(s', a')$). When generalizing to multiple agents, s and a become vectors of the state and action of every agent and the max is replaced by a prediction of the behavior of other agents. Different algorithms use different predictions; for example, Nash-Q uses a Nash equilibrium calculation [42]. See [66] for a survey.

Unfortunately, these algorithms converge too slowly for a large distributed system. The algorithm needs to experience each possible action profile many times to guarantee convergence. So, with n agents and k strategies, the naive convergence time is $O(k^n)$. Even with a better representation for anonymous games, the convergence time is still $O(n^k)$ (typically $k \ll n$). There is also a more fundamental problem with this approach: it assumes information that an agent is unlikely to have. In order to know which value to update, the agent must learn the action chosen by every other agent. In practice, an agent will learn something about the actions of the agents with whom he directly interacts, but

is unlikely to gain much information about the actions of other agents.

Another approach is *no-regret learning*, where agents choose a strategy for each round that guarantees that the regret of their choices will be low. Hart and Mas-Colell [36] present such a learning procedure that converges to a *correlated equilibrium*² given knowledge of what the payoffs of every action would have been in each round. They also provide a variant of their algorithm that requires only information about the agent's actual payoffs [37]. However, to guarantee convergence to within ϵ of a correlated equilibrium requires $O(kn/\epsilon^2 \log kn)$, still too slow for large systems. Furthermore, the convergence guarantee is that the distribution of play converges to equilibrium; the strategies of individual learners will not converge. Many other no-regret algorithms exist [11]. In Section 4.3, we use the Exp3 algorithm [5]. They can achieve even better convergence in restricted settings. For example, Blum et al. [10] showed that in routing games a continuum of no-regret learners will approximate Nash equilibrium in a finite amount of time.

Foster and Young [25] use a stage-learning procedure that converges to Nash equilibrium for two-player games. Germano and Lugosi [30] showed that it converges for generic n -player games (games where best replies are unique). Young [81] uses a similar algorithm without explicit stages that also converges for generic n -player games. Rather than selecting best replies, in these algorithms agents choose new actions randomly when not in equilibrium. Unfortunately, these algorithms involve searching the whole strategy space, so their convergence time is exponential. Another algorithm that uses stages to provide a stable learning environment is the ESRL algorithm for coordinated ex-

²Correlated equilibrium is a more general solution concept than Nash equilibrium; see [60]; every Nash equilibrium is a correlated equilibrium, but there may be correlated equilibria that are not Nash equilibria.

ploration [74].

Marden et al. [55, 54] use an algorithm with experimentation and best replies but without explicit stages that converges for *weakly acyclic games*, where best-reply dynamics converge when agents move one at a time, rather than moving all at once, as we assume here. Convergence is based on the existence of a sequence of exploration moves that lead to equilibrium. With n agents who explore with probability ϵ , this analysis gives a convergence time of $O(1/\epsilon^n)$. Furthermore, the guarantee requires ϵ to be sufficiently small that agents essentially explore one at a time, so ϵ needs to be $O(1/n)$.

There is a long history of work examining simple learning procedures such as *fictitious play* [28], where each agent makes a best response assuming that each other player's strategy is characterized by the empirical frequency of his observed moves. In contrast to algorithms with convergence guarantees for general games, these algorithms fail to converge in many games. But for classes of games where they do converge, they tend to do so rapidly. However, most work in this area assumes that the actions of agents are observed by all agents, agents know the payoff matrix, and payoffs are deterministic. A recent approach in this tradition is based on the Win or Learn Fast principle, which has limited convergence guarantees but often performs well in practice [13]. Hopkins [41] showed that many such procedures converge in symmetric games with an infinite number of learners, although his results provide no guarantees about the rate of convergence.

There is also a body of empirical work on the convergence of learning algorithms in multiagent settings. Q-learning has had empirical success in pricing games [71], n -player cooperative games [18], and grid world games [12]. Green-

wald et al. [32] showed that a number of algorithms, including stage learning, converge in a variety of simple games. Marden et al. [54] found that their algorithm converged much faster in a congestion game than the theoretical analysis would suggest. Our theorem suggests an explanation for these empirical observations: best-reply dynamics converge in all these games. While our theorem applies directly only to stage learning, it provides intuition as to why algorithms that learn “quickly enough” and change their behavior “slowly enough” rapidly converge to Nash equilibrium in practice.

CHAPTER 2

A MODEL OF A SCRIP SYSTEM

2.1 The Model

Before specifying our model formally, we give an intuitive description of what our model captures. We model a scrip system where, as in the babysitting co-op, agents provide each other with service. There is a single service (babysitting) that agents occasionally want. In practice, at any given time, a number of agents will want service, but to simplify the formal description and analysis we model the scrip system as proceeding in a series of rounds where, in each round, a single agent wants service (the time between rounds will be adjusted to capture the growth in parallelism as the number of agents grows).¹ In each round, after an agent requests service, other agents have to decide whether they want to volunteer to provide service. However, not all agents may be able to satisfy the request (not everyone can babysit every night). While, in practice, the ability of agents to provide service at various times may be correlated for a number of reasons (some agents might have very young children that only certain agents are willing to babysit; being unavailable in one round might be correlated with being unavailable in the next round; and so on), for simplicity we model the ability to provide service using a single probability, and assume that the events of an agent being able to provide service in different rounds or two agents being able to provide service in the same or different rounds are independent. If there is at least one volunteer, someone is chosen from among the volunteers

¹For large numbers of agents, our model converges to one in which agents make requests in real time, and the time between an agent's requests is exponentially distributed. In addition, the time between requests served by a single player is also exponentially distributed.

(uniformly at random) to satisfy the request. The requester then gains some utility (he was able to go out because he had a babysitter) and the volunteer loses some utility (people would rather do something other than babysit), and the requester pays the volunteer a fee that we fix at one dollar. As is standard in the literature, we assume that agents discount future payoffs. This captures the intuition that a util now is worth more than a util tomorrow, and allows us to compute the total utility derived by an agent in an infinite game. The amount of utility gained by having a service performed and the amount lost by performing it, as well as many other parameters may depend on the agent.

More formally, we assume that agents have a *type* t drawn from some finite set T of types. We can describe the entire population of agents using the pair (T, \vec{f}) , where \vec{f} is a vector of length $|T|$ and f_t is the fraction with type t . For most of the paper, we consider only what we call *standard agents*. These are agents who derive no pleasure from performing a service, and for whom money has no intrinsic value. We can characterize the type of a standard agent by a tuple $t = (\alpha_t, \beta_t, \gamma_t, \delta_t, \rho_t, \chi_t)$, where

- $\alpha_t > 0$ reflects the cost for an agent of type t to satisfy a request;
- $0 < \beta_t \leq 1$ is the probability that an agent of type t can satisfy a request;
- $\gamma_t > \alpha_t$ is the utility that an agent of type t gains for having a request satisfied;
- $0 < \delta_t < 1$ is the rate at which an agent of type t discounts utility;
- $\rho_t > 0$ represents the (relative) request rate (some people need babysitting more often than others). For example, if there are two types of agents with $\rho_{t_1} = 2$ and $\rho_{t_2} = 1$ then agents of the first type will make requests twice as often as agents of the second type. Since these request rates are relative,

we can multiply them all by a constant to normalize them. To simplify later notation, we assume the ρ_t are normalized so that $\sum_{t \in T} \rho_t f_t = 1$; and

- $\chi_t > 0$ represents the (relative) likelihood of an agent to be chosen when he volunteers (some babysitters may be more popular than others). In particular, this means the relative probability of two given agents being chosen is independent of which other agents volunteer.
- $\omega_t = \beta_t \chi_t / \rho_t$ is not part of the tuple, but is an important derived parameter that, as we will see in Section 2.2, helps determine how much money an agent will have.

Representing the population of agents in a system as (T, \vec{f}) captures the essential features of a scrip system we want to model: there are a large number of agents who may have different types. Note that some tuples (T, \vec{f}) may be incompatible with there being some number N of agents. For example, if there are two types, and \vec{f} says that half of the agents are of each type, then there cannot be 101 agents. Similar issues arise when we want to talk about the amount of money in example, by specifying how to assign to types to agents in a way that we could deal with this problem in a number of ways (for example, by having each agent determine his type at random according to the distribution \vec{f}). For convenience, we simply do not consider population sizes that are incompatible with \vec{f} . This is the approach used in the literature on *N-replica economies* [56].

Formally, we consider games specified by a tuple (T, \vec{f}, h, m, n) , where T and \vec{f} are as defined above, $h \in \mathbb{N}$ is the *base* number of agents of each type, $n \in \mathbb{N}$ is number of *replicas* of these agents and $m \in \mathbb{R}^+$ is the average amount of money. The total number of agents is thus hn . We ensure that the number of agents of type t is exactly f_t and that the average amount of money is exactly

m by requiring that $f_1 h \in \mathbb{N}$ and $mh \in \mathbb{N}$. Having created a base population satisfying these constraints, we can make an arbitrary number of copies of it. More precisely, we assume that agents $0 \dots f_1 h - 1$ have type t_1 , agents $f_1 h \dots (f_1 + f_2)h - 1$ have type t_2 , and so on through agent $h - 1$. These base agents determine the types of all other agents. Each agent $j \in \{h, \dots, hn - 1\}$ has the same type as $j \bmod h$; that is, all the agents of the form $j + kh$ for $k = 1, \dots, n - 1$ are replicas of agent j . At the start of the game, we initially allocate each of the hmn dollars in the system to an agent chosen uniformly at random.

We now describe (T, \vec{f}, h, m, n) as an infinite extensive-form game. A non-root node in the game tree is associated with a round number (how many requests have been made so far), a phase number, either 1, 2, 3, or 4 (which describes how far along we are in determining the results of the current request), a vector \vec{x} where x_i is the current amount of money agent i has, and $\sum_i x_i = mhn$, and, for some nodes, some additional information whose role will be made clear below. We use $\tau(i)$ to denote the type of agent i .

- The game starts at a special root node, denoted Λ , where nature moves. Intuitively, at Λ , nature allocates money uniformly at random, so it transitions to a node of the form $(0, 1, \vec{x})$: round zero, phase one, and allocation of money \vec{x} , and each possible transition is equally likely.
- At a node of the form $(r, 1, \vec{x})$, nature selects an agent to make a request in the current round. Agent i is chosen with probability $\rho_{\tau(i)}/hn$. If i is chosen, a transition is made to $(r, 2, \vec{x}, i)$.
- At a node of the form $(r, 2, \vec{x}, i)$, nature selects the set V of agents (not including i) able to satisfy the request. Each agent $j \neq i$ is included in V with probability $\beta_{\tau(j)}$. If V is chosen, a transition is made to $(r, 3, \vec{x}, i, V)$.

- At a node of the form $(r, 3, \vec{x}, i, V)$, each agent in V chooses whether to volunteer. If V' is the set of agents who choose to volunteer, a transition is made to $(r, 4, \vec{x}, i, V')$.
- At a node of the form $(r, 4, \vec{x}, i, V')$, if $V' \neq \emptyset$, nature chooses a single agent in V' to satisfy the request. Each agent j is chosen with probability $\chi_{\tau(j)} / \sum_{j' \in V'} \chi_{\tau(j')}$. If j is chosen, a transition is made to $(r + 1, 1, \vec{x})$, where

$$x'_j = \begin{cases} x_j - 1 & \text{if } i = j \text{ and } x_j > 0, \\ x_j + 1 & \text{if } j \text{ is chosen by nature and } x_i > 0, \\ x_j & \text{otherwise.} \end{cases}$$

If $V' = \emptyset$, nature has no choice; a transition is made to $(r + 1, 1, \vec{x})$ with probability 1.

A strategy for agent j describes whether or not he will volunteer at every node of the form $(r, 3, \vec{x}, i, V)$ such that $j \in V$. (These are the only nodes where j can move.) A strategy profile \vec{S} consists of one strategy per agent. A strategy profile \vec{S} determines a probability distribution over paths $\Pr_{\vec{S}}$ in the game tree. Each path determines the value of the following random two variables:

- x_i^r , the amount of money agent i has during round r , defined as the value of x_i at the nodes with round number r and
- u_i^r , the utility of agent i for round r . If i is a standard agent, then

$$u_i^r = \begin{cases} \gamma_{\tau(i)} & \text{if a node } (r, 4, \vec{x}, i, V') \text{ is on the path with } V' \neq \emptyset \\ -\alpha_{\tau(i)} & \text{if } i \text{ is chosen by nature at node } (r, 4, \vec{x}, j, V') \\ 0 & \text{otherwise.} \end{cases}$$

$U_i(\vec{S})$, the total expected utility of agent i if strategy profile \vec{S} is played is the discounted sum of his per round utilities u_i^r , but the exact form of the discounting requires some explanation. As the number of agents increases, we would expect more requests to be made per unit time, and the expected number of requests an agent makes per unit time to be constant. Since only one agent makes a request per round, it seems that a reasonable way to model this is to take the time between rounds to be $1/n$, where n is the number of agents. The discount rate—which can be thought of as the present value of getting one util one round in the future—has to be modified as well. It turns out that the obvious choice of discount rate, $\delta_i^{1/n}$, is not appropriate. To understand why, consider an agent who has all of his requests satisfied. When there are just h agents, he is chosen to make a request each round with probability ρ_i/h . His total expected utility with a discount rate of δ is $\sum_{r=0}^{\infty} \delta^r \rho_i \gamma_i / h = (\rho_i \gamma_i / h) / (1 - \delta_i)$. With n replicas, scaling the discount rate as $\delta_i^{1/n}$ gives $\sum_{r=0}^{\infty} \delta_i^{r/n} \rho_i \gamma_i / (hn) = (\rho_i \gamma_i / (hn)) / (1 - \delta_i^{1/n})$. Thus, using this scaling, the agent's utility for having all his requests satisfied decreases as n increases. This seems unnatural. If we instead use the discount rate $(1 - (1 - \delta_i)/n)$, his expected utility is $\sum_{r=0}^{\infty} (1 - (1 - \delta_i)/n)^r (\rho_i \gamma_i / (hn)) = (\rho_i \gamma_i / (hn)) / (1 - (1 - (1 - \delta_i)/n)) = (\rho_i \gamma_i / h) / (1 - \delta_i)$, which is independent of n , and seems much more reasonable.

Using the discount rate $(1 - (1 - \delta_i)/n)$ solves one problem, but leaves another. A larger δ_i is meant to reflect a more patient agent, who gives future utility a higher weight. However, as the preceding equation shows, increasing δ_i also increases total utility. To counteract this, we multiply the total discounted sum by $(1 - \delta_i)$. This is standard in economics, for example in the folk theorem for repeated games [29]. With these considerations in mind, the total expected

utility of agent i given the vector of strategies \vec{S} is

$$U_i(\vec{S}) = (1 - \delta_{\tau(i)}) \sum_{r=0}^{\infty} (1 - (1 - \delta_{\tau(i)})/n)^r E_{\vec{S}}[u_i^r], \quad (2.1)$$

In modeling the game this way, we have implicitly made a number of assumptions. For example, we have assumed that all of agent i 's requests that are satisfied give agent i the same utility, and that prices are fixed. We discuss the implications of these assumptions in Chapter 5.

Our solution concept is the standard notion of an approximate Nash equilibrium. As usual, given a strategy profile \vec{S} and agent i , we use (S'_i, \vec{S}_{-i}) to denote the strategy profile that is identical to \vec{S} except that agent i uses S'_i .

Definition 2.1.1. A strategy S'_i for agent i is an ϵ -best reply to a strategy profile \vec{S}_{-i} for the agents other than i in the game (T, \vec{f}, h, m, n) if, for all strategies S''_i ,

$$U_i(S''_i, \vec{S}_{-i}) \leq U_i(S'_i, \vec{S}_{-i}) + \epsilon.$$

Definition 2.1.2. A strategy profile \vec{S} for the game (T, \vec{f}, h, m, n) is an ϵ -Nash equilibrium if for all agents i , S_i is an ϵ -best reply to \vec{S}_{-i} . A Nash equilibrium is an epsilon-Nash equilibrium with $\epsilon = 0$.

As we show in Section 2.3, (T, \vec{f}, h, m, n) has equilibria where agents use a particularly simple type of strategy, called a *threshold strategy*. Intuitively, an agent with “too little” money will want to work, to minimize the likelihood of running out due to making a long sequence of requests before being able to earn more money. On the other hand, a rational agent with plenty of money will think it is better to delay working, thanks to discounting. These intuitions suggest that the agent should volunteer if and only if he has less than a certain amount of money. Let s_k be the strategy where an agent volunteers if and only if

the requester has at least 1 dollar and the agent has less than k dollars. Note that s_0 is the strategy where the agent never volunteers. While everyone playing s_0 is a Nash equilibrium (nobody can do better by volunteering if no one else is willing to), it is an uninteresting one.

We frequently consider the situation where each agent of type t uses the same threshold s_{k_t} . In this case, a single vector \vec{k} suffices to indicate the threshold of each type, and we can associate with this vector the strategy $\vec{S}(\vec{k})$ where $\vec{S}(\vec{k})_i = s_{k_{\tau(i)}}$ (i.e., agent i of type $\tau(i)$ uses threshold $k_{\tau(i)}$).

For the rest of this paper, we focus on threshold strategies (and show why it is reasonable to do so). When we consider the threshold strategy $\vec{S}(\vec{k})$, for ease of exposition, we assume in our analysis that $mhn < \sum_i f_i k_i hn$. To understand why, note that mhn is the total amount of money in the system. If $mhn \geq \sum_i f_i k_i hn$, then if the agents use a threshold $\vec{S}(\vec{k})$, the system will quickly reach a state where each agent has k_i dollars, so no agent will volunteer. This is equivalent to all agents using a threshold of 0, and similarly uninteresting.

2.2 Analyzing the Distribution of Wealth

Our main goal is to show that there exists an approximate equilibrium where all agents play threshold strategies. In this section, we examine a more basic question: if all agents play a threshold strategy, what happens? We show that there is some distribution over money (i.e., a distribution that describes what fraction of people have each amount of money) such that the system “converges” to this distribution in a sense to be made precise shortly. In addition to providing an understanding of system behavior that underpins our later results, this result

also provides a strong guarantee about the stability of the economy.

Suppose that all agents of each type t use the same threshold k_t , so we can write the vector of thresholds as \vec{k} . For simplicity, assume that each agent has at most k_t dollars. We can make this assumption with essentially no loss of generality, since if someone has more than k_t dollars, he will just spend money until he has at most k_t dollars. After this point he will never acquire more than k_t . Thus, eventually the system will be in a state where, for all types t , no agent of type t has more than k_t dollars.

We are interested in the vectors \vec{x}^r that can be observed in round r (recall that x_i^r is the amount of money that agent i has at round r). By assumption, if agent i has type $\tau(i)$, then $x_i^r \in \{0, \dots, k_{\tau(i)}\}$. In addition, since the total amount of money is hmn ,

$$\vec{x}^r \in X_{T, \vec{f}, h, m, n, \vec{k}} = \{\vec{x} \in \mathbb{N}^{hm} \mid \forall i. x_i \leq k_{\tau(i)}, \sum_i x_i = hmn\}.$$

The evolution of \vec{x}^r can be described by a Markov chain $\mathcal{M}_{T, \vec{f}, h, m, n, \vec{k}}$ over the state space $X_{T, \vec{f}, h, m, n, \vec{k}}$. For brevity, we refer to the Markov chain and state space as \mathcal{M} and X , respectively, when the subscripts are clear from context. It is possible to move from state s to state s' in a single round if, by choosing a particular agent i to make a request and another agent j to satisfy it, i 's amount of money in s' is 1 more than in s ; j 's amount of money in s' is 1 less than in s , and all other agents have the same amount of money in s and s' . Therefore, the probability of a transition from a state \vec{x} to \vec{y} is 0 unless there exist two agents i and j such that $\vec{y}_{i'} = \vec{x}_{i'}$ for all $i' \notin \{i, j\}$, $\vec{y}_i = \vec{x}_i + 1$, and $\vec{y}_j = \vec{x}_j - 1$. In this case, the probability of transitioning from \vec{x} to \vec{y} is the probability of j being chosen to make a request and i being chosen to satisfy it. Let $\Delta_{\vec{f}, m, \vec{k}}$ denote the set of probability distributions d on $\cup_{t \in T} \{t\} \times \prod_t \{0, \dots, k_t\}$ such that for all types

t , $\sum_{l=0}^{k_t} d(t, l) = f_t$. We can think of $d(t, l)$ as the fraction of agents of type t that have l dollars. We can associate each state \vec{x} with its corresponding distribution $d^{\vec{x}}$. This is a useful way of looking at the system, since we typically just care about the fraction of people with each amount of money, not the amount that each particular agent has. We show that, if n is large, then there is a distribution $d^* \in \Delta_{\vec{f}, m, \vec{k}}$ such that, after a sufficient amount of time, the Markov chain \mathcal{M} is almost always in a state \vec{x} such that $d^{\vec{x}}$ is close to d^* . Thus, agents can base their decisions about what strategy to use on the assumption that they will be in a state where the distribution of money is essentially d^* .

We can in fact completely characterize the distribution d^* . Given two distributions $d, q \in \Delta_{\vec{f}, m, \vec{k}}$, let

$$H(d||q) = - \sum_{\{(t,j): q(t,j) \neq 0\}} d(t, j) \log d(t, j)/q(t, j)$$

denote the *relative entropy* of d relative to q ($H(d||q) = \infty$ if $d(t, j) = 0$ and $q(t, j) \neq 0$ or vice versa); this is also known as the *Kullback-Leibler divergence of q from d* [20]. If Δ is a closed convex set of distributions, then it is well known that, for each q , there is a unique distribution in Δ that minimizes the entropy relative to q . Since $\Delta_{\vec{f}, m, \vec{k}}$ is easily seen to be a closed convex set of distributions, in particular, this is the case for $\Delta_{\vec{f}, m, \vec{k}}$. We now show that there exists a q such that, for n sufficiently large, the Markov chain \mathcal{M} is almost always in a state \vec{x} such that $d^{\vec{x}}$ is close to the distribution $d_{q, \vec{f}, m}^* \in \Delta_{\vec{f}, m, \vec{k}}$ that minimizes entropy relative to q . (We omit some or all of the subscripts on d^* when they are not relevant.) The statement is correct under a number of senses of “close”. For definiteness, we consider the Euclidean distance. Given $\varepsilon > 0$ and q , let $X_{T, \vec{f}, h, m, n, \vec{k}, \varepsilon, q}$ (or $X_{\varepsilon, q}$ for brevity) denote the set of states $\vec{x} \in X_{T, \vec{f}, h, m, n, \vec{k}}$ such that $\sum_{(t,j)} |d^{\vec{x}}(t, j) - d_q^*(t, j)|^2 < \varepsilon$.

Let $I_{q, n, \varepsilon}^r$ be the random variable that is 1 if $d^{\vec{x}^r} \in X_{\varepsilon, q}$, and 0 otherwise.

Theorem 2.2.1. *For all games $(T, \vec{f}, h, m, 1)$, all vectors \vec{k} of thresholds, and all $\varepsilon > 0$, there exist $q \in \Delta_{\vec{f}, m, \vec{k}}$ and n_ε such that, for all $n > n_\varepsilon$, there exists a round r^* such that, for all $r > r^*$, we have $\Pr(I_{q, n, \varepsilon}^r = 1) > 1 - \varepsilon$.*

The proof of Theorem 2.2.1 can be found in Appendix A.1. One interesting special case of the theorem is when there exist β, χ , and ρ such that for all types t , $\beta_t = \beta$, $\chi_t = \chi$, and $\rho_t = \rho$. In this case q is the distribution $q(t, j) = f_t/(k_t + 1)$ (i.e., q is uniform within each type t). We sketch the proof for this special case here.

Proof. (Sketch) Using standard techniques, we can show that our Markov Chain has a *limit distribution* π such that for all \vec{y} , $\lim_{r \rightarrow \infty} \Pr(\vec{x}^r = \vec{y}) = \pi(\vec{y})$. Let $T_{\vec{x}\vec{y}}$ denote the probability of transitioning from state \vec{x} to state \vec{y} . It is easily verified by an explicit computation of the transition probabilities that (in this special case) $T_{\vec{x}\vec{y}} = T_{\vec{y}\vec{x}}$. It is well known that this symmetry implies that π is the uniform distribution [64]. Thus, after a sufficient amount of time, the distribution of \vec{x}^r will be arbitrarily close to uniform.

Since, for large r , $\Pr(\vec{x}^r = \vec{y})$ is approximately $1 / |X|$, the probability of \vec{x}^r being in a set of states is the size of the set divided by the total number of states. Using a straightforward combinatorial argument, it can be shown that the fraction of states not in $X_{\varepsilon, q}$ is bounded by $p(n)/e^{cn}$, where p is a polynomial. This fraction goes to 0 as n gets large. Thus, for sufficiently large n , $\Pr(I_{q, n, \varepsilon}^r = 1) > 1 - \varepsilon$. \square

The last portion of the proof sketch is actually a standard technique from statistical mechanics that involves showing that there is a *concentration phenomenon* around the maximum entropy distribution [46]. In this special case, when π is the uniform distribution, the number of states corresponding to a particular distribution d is proportional to $e^{nH(d)}$ (where H here is the standard entropy

function). In general, each state is not equally likely, which is why the general proof in Appendix A.1 uses relative entropy.

Theorem 2.2.1 tells us that, after enough time, the distribution of money is almost always close to some d^* , where d^* can be characterized as a distribution that minimizes relative entropy subject to some constraints. Let $q(t, i) = (\omega_t)^i / (\sum_t \sum_{j=0}^{k_t} (\omega_t)^j)$. Then the value of d^* is given by the following lemma.

Lemma 2.2.1.

$$d^*(t, i) = \frac{f_t \lambda^i q(t, i)}{\sum_{j=0}^{k_t} \lambda^j q(t, j)}, \quad (2.2)$$

where λ is the unique value such that

$$\sum_t \sum_i i d^*(t, i) = m. \quad (2.3)$$

The proof of Lemma 2.2.1 is omitted because it can be easily checked using Lagrange multipliers in the manner of [46] where the function to be minimized is the entropy of d^* relative to q and the constraints are there a f_t fraction of the agents are of type t and the average amount of money is m .

2.3 Existence of Equilibria

We have seen that the system is well behaved if the agents all follow a threshold strategy; we now want to show that, if the discount factor δ is sufficiently large for all agents, there is a nontrivial approximate Nash equilibrium where they do so (that is, an approximate Nash equilibrium where all the agents use s_k for some $k > 0$.) To understand why we need δ to be sufficiently large, note that if δ is small, then agents have no incentive to work. Intuitively, if future utility

is sufficiently discounted, then all that matters is the present, and there is no point in volunteering to work. Thus, for sufficiently small δ , s_0 is the only equilibrium. To show that there is a nontrivial equilibrium if the discount factor is sufficiently large, we first show that, if every other agent is playing a threshold strategy, then there is an approximate best reply that is also a threshold strategy. Furthermore, we show that the best-reply function is monotone; that is, if some agents change their strategy to one with a higher threshold, no other agent can do better by lowering his threshold. This makes our game one with what Milgrom and Roberts [57] call *strategic complementarities*. Using results of Tarski [70], Topkis [73] showed that there are pure strategy equilibria in such games, since the process of starting with a strategy profile where everyone always volunteers (i.e., the threshold is ∞) and then iteratively computing the best-reply profile to it converges to a Nash equilibrium in pure strategies. This procedure also provides an efficient algorithm for explicitly computing equilibria.

To see that threshold strategies are approximately optimal, consider a single agent i of type t and fix the vector \vec{k} of thresholds used by the other agents. If we assume that the number of agents is large, what an agent i does has essentially no affect on the behavior of the system (although it will, of course, affect that agent's payoffs). In particular, this means that the distribution q of Theorem 2.2.1 characterizes the distribution of money in the system. This distribution, together with the vector \vec{k} of thresholds, determines what fraction of agents volunteers at each step. This, in turn, means that from the perspective of agent i , the problem of finding an optimal response to the strategies of the other agents reduces to finding an optimal policy in a Markov decision process (MDP) $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$. The behavior of the MDP $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$ depends on two probabilities: p_u and

p_d . Informally, p_u is the probability of i earning a dollar during each round if i is willing to volunteer, and p_d is the probability that i will be chosen to make a request during each round. Note that p_u depends on m, \vec{k} , and t (although it turns out that p_d depends only on n , the number of agents in the system) and t ; if the dependence of p_u on m, \vec{k} , and/or t is important, we add the relevant parameters to the superscript, writing, for example, $p_u^{m, \vec{k}}$. We show that the optimal policy for i in $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$ is a threshold policy, and that this policy is an ε -optimal strategy for G . Importantly, the same policy is optimal independent of the value of n . This allows us to ignore the exact size of the system in our later analysis.

For many of our later results and discussion, it will be important to understand how p_u , p_d , and t affect the optimal policy for $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$ and thus the ε -optimal strategies in the game. We use this understanding in this section to show that there exist nontrivial equilibria in Lemma 2.3.3, to show that adding money increases social welfare in Section 3.1, to understand how agent behaviors affect social welfare in Section 3.2, and to identify agent types from their behavior in Section 3.3.

In the following lemma, whose proof (and the relevant formal definitions) are deferred to Appendix A.2, Equation (2.4), quantifies the effects of these parameters. When choosing whether he should volunteer with his current amount of money, an agent faces a choice of whether to pay a utility cost of α_t now in exchange for a discounted payoff of γ_t when he eventually spends the resulting dollar. His choice will depend on how much time he expects to pass before he spends that dollar, which in turn depends on his current amount of money k and the probabilities p_u and p_d . The following lemma quantifies this calculation.

Lemma 2.3.1. *Consider the games $G_n = (T, \vec{f}, h, m, n)$ (where T, \vec{f}, h , and m are fixed,*

but n may vary). There exists a k such that for all n , s_k is an optimal policy for $\mathcal{P}_{G_n, \vec{S}(\vec{k}), t}$. The threshold k is the maximum value of κ such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u, p_d)}] \gamma_t, \quad (2.4)$$

where $J(\kappa, p_u, p_d)$ is a random variable whose value is the first round in which an agent starting with κ dollars, using strategy s_κ , and with probabilities p_u and p_d of earning a dollar and of being chosen given that he volunteers, respectively, runs out of money.

The following theorem shows that an optimal threshold policy for $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ is an ε -optimal strategy for G . In particular, this means that Equation (2.4) allows us to understand how changing parameters affect an ε -optimal strategy for G , not just for $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$.

Theorem 2.3.1. *For all games $G = (T, \vec{f}, h, m, n)$, all vectors \vec{k} of thresholds, and all $\varepsilon > 0$, there exist n_ε^* and $\delta_{\varepsilon, n}^*$ such that for all $n > n_\varepsilon^*$, types $t \in T$, and $\delta_t > \delta_{\varepsilon, n}^*$, an optimal threshold policy for $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ is an ε -best reply to the strategy profile $\vec{S}(\vec{k})_{-i}$ for every agent i of type t .*

We defer the proof of Theorem 2.3.1 to Appendix A.2. While, in this and later theorems, the acceptable values of $\delta_{\varepsilon, n}^*$ depend on n , they are independent if, as we suggest in Section 2.4, the Markov Chain from Section 2.2 is rapidly mixing.

Given a game $G = (T, \vec{f}, h, m, n)$ and a vector \vec{k} of thresholds, Lemma 2.3.1 gives an optimal threshold k'_t for each type t . Theorem 2.3.1 guarantees that $s_{k'_t}$ is an ε -best reply to $\vec{S}_{-i}(\vec{k})$, but does not rule out the possibility of other best replies. However, for ease of exposition, we will call k'_t the best reply to \vec{S}_{-i} and call $BR_G(\vec{k}) = \vec{k}'$ the best-reply function. The following lemma shows that this function is monotone: it is non-decreasing in \vec{k} and non-increasing in m . Along

the way, we prove that several other quantities are monotone. First, we show that $\lambda_{m,\vec{k}}$, the value of λ from Lemma 2.2.1 given m and \vec{k} , is non-decreasing in m and non-increasing in \vec{k} . We use this to show that $p_u^{m,\vec{k}}$ is non-increasing in \vec{k} and non-decreasing in m . We defer the proof to Appendix A.2.

Lemma 2.3.2. *Consider the family of games $G_m = (T, \vec{f}, h, m, n)$ and the strategies $\vec{S}(\vec{k})$, for $mhn < \sum_i f_i k_i hn$. For this family of game, $\lambda_{m,\vec{k}}$ is non-decreasing in m and non-increasing in \vec{k} ; $p_u^{m,\vec{k}}$ is non-decreasing in m and non-increasing in \vec{k} ; and the function BR_G is non-decreasing in \vec{k} and non-increasing in m .*

Monotonicity is enough to guarantee the existence of an equilibrium. Unfortunately, we know that a trivial equilibrium always exists in threshold strategies: all agents choose a threshold of \$0, so no agent ever volunteers. To guarantee the existence of a nontrivial equilibrium, it is sufficient to show there is some vector \vec{k} of thresholds such that $BR_G(\vec{k}) > \vec{k}$. The following lemma, whose proof is again deferred to Appendix A.2, shows that we can always find such a point for sufficiently large δ_t .

Lemma 2.3.3. *For all games $G = (T, \vec{f}, h, m, n)$, there exists a $\delta^* < 1$ such that if $\delta_t > \delta^*$ for all t , there is a vector \vec{k} of thresholds such that $BR_G(\vec{k}) > \vec{k}$.*

We are now ready to prove our main theorem: there exists a non-trivial equilibrium where all agents play threshold strategies greater than zero.

Theorem 2.3.2. *For all games $G = (T, \vec{f}, h, m, 1)$ and all ϵ , there exist n_ϵ^* and $\delta_{\epsilon,n}^*$ such that, if $n > n_\epsilon^*$ and $\delta_t > \delta_{\epsilon,n}^*$ for all t , then there exists a nontrivial vector \vec{k} of thresholds that is an ϵ -Nash equilibrium. Moreover, there exists a greatest such vector.*

Proof. By Lemma 2.3.2, BR_G is a non-decreasing function on a complete lattice, so Tarski's fixed point theorem [70] guarantees the existence of a greatest and

least fixed point; these fixed points are equilibria. The least fixed point is the trivial equilibrium. We can compute the greatest fixed point by starting with the strategy profile (∞, \dots, ∞) (where each agent uses the strategy S_∞ of always volunteering) and considering ϵ -best-reply dynamics, that is, iteratively computing the ϵ -best-reply strategy profile. Monotonicity guarantees this process converges to the greatest fixed point, which is an equilibrium (and is bound to be an equilibrium in pure strategies, since the best reply is always a pure strategy). Since there is a finite amount of money, this process needs to be repeated only a finite number of times. By Lemma 2.3.3, there exists a \vec{k} such that $BR_G(\vec{k}) > \vec{k}$. Monotonicity then guarantees that $BR_G(BR_G(\vec{k})) \geq BR_G(\vec{k})$ and similarly for any number of applications of BR_G . If \vec{k}^* is the greatest fixed point of BR_G , then $\vec{k}^* > \vec{k}$. Thus, the greatest fixed point is a nontrivial equilibrium. \square

The proof of Theorem 2.3.2 also provides an algorithm for finding equilibria that seems efficient in practice: start with the strategy profile (∞, \dots, ∞) and iterate the best-reply dynamics until an equilibrium is reached.

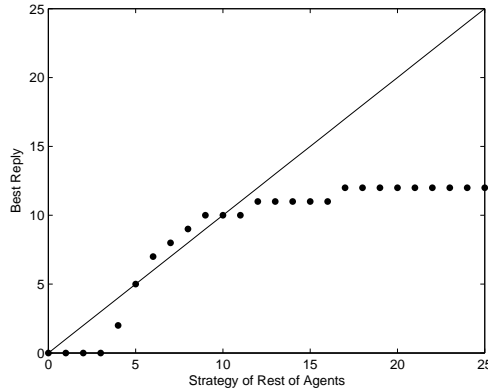


Figure 2.1: A hypothetical best-reply function with one type of agent.

There is a subtlety in our results. In general, there may be many equilibria. From the perspective of social welfare, some will be better than others. As

we show in Section 3.1, strategies that use smaller (but nonzero) thresholds increase social welfare. Consider the best-reply function shown in Figure 2.1. In the game G in the example, there is only one type of agent, so $BR_G : \mathbb{N} \rightarrow \mathbb{N}$. In equilibrium, we must have $BR(k) = k$; that is, an equilibrium is characterized by a point on the line $y = x$. This example has three equilibria, where all agents play s_0 , s_5 , and s_{10} respectively. The strategy profile where all agents play s_5 is the equilibrium that maximizes social welfare, while s_{10} is the greatest equilibrium.

In the rest of this paper, we focus on the greatest equilibrium in all our applications (although a number of our results hold for all nontrivial equilibria). This equilibrium has several desirable properties. First, it is guaranteed to be stable; best-reply dynamics from nearby points converge to it. By way of contrast, best-reply dynamics moves the system away from the equilibrium S_5 in Figure 2.1. Unstable equilibria are difficult to find in practice, and seem unlikely to be maintained for any length of time. Second, the “greatest” equilibrium is the one found by the natural algorithm given in Theorem 2.3.2. The proof of the theorem shows that it is also the outcome that will occur if agents adopt the reasonable initial strategy of starting with a large threshold and then using best-reply dynamics. Finally, by focusing on the worst nontrivial equilibrium, our results provide guarantees on social welfare, in the same way that results on *price of anarchy* [65] provide guarantees (since price of anarchy considers the social welfare of the Nash equilibrium with the worst social welfare).

2.4 Simulations

Theorem 2.2.1 proves that, for a sufficiently large number n of agents, and after a sufficiently large number r of rounds, the distribution of wealth will almost always be close to the distribution that minimizes relative entropy. In this section, we simulate the game to gain an understanding of how large n and r need to be in practice. The simulations show that our theoretical results apply even to relatively small systems; we get tight convergence with a few thousand agents, and weaker convergence for smaller numbers, in very few rounds rounds, indeed, a constant number per agent.

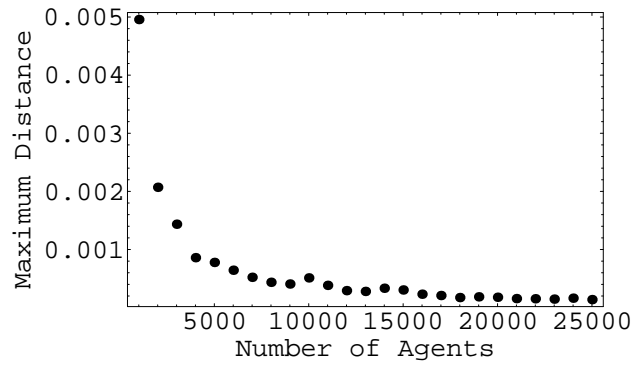


Figure 2.2: Maximum distance from minimum relative entropy distribution over 10^6 timesteps.

The first simulation explores the tightness of convergence to the distribution that minimizes relative entropy for various values of n . We used a single type of agent, with $\beta = \rho = \chi = 1$, $m = 2$, and $k = 5$. For each value of n , the simulation was started with a distribution of money as close as possible to the distribution d^* that minimizes relative entropy to the distribution q defined in Theorem 2.2.1 that characterizes the distribution of money in equilibrium (when the threshold strategy 5 is used). We then computed the maximum Euclidean distance be-

tween d^* and the observed distribution over 10^6 rounds. As Figure 2.2 shows, the system does not move far from d^* once it is there. For example, if $n = 5000$, the system is never more than distance .001 from d^* . If $n = 25,000$, it is never more than .0002 from d^* .

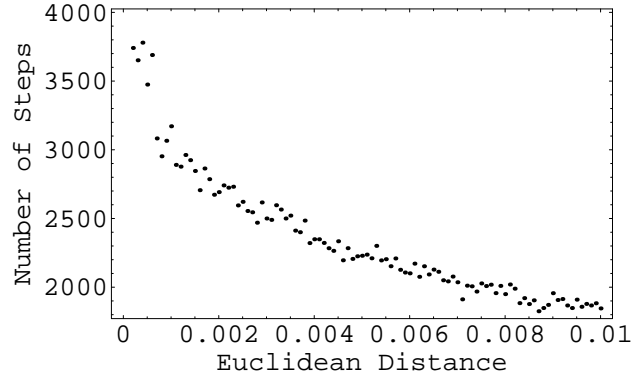


Figure 2.3: Distance from minimum relative entropy distribution with 1000 agents.

Figure 2.2 does show a larger distance for $n = 1000$, although in absolute terms it is still small. The next simulation shows that, while the system may occasionally move away from d^* , it quickly converges back to it. We averaged 10 runs of the Markov chain, starting from an extreme distribution (every agent has either \$0 or \$5), and considered the average time needed to come within various distances of d^* . As Figure 2.3 shows, after 2 rounds per agent, on average, the Euclidean distance from the average distribution of money to d^* is .008; after 3 rounds per agent, the distance is down to .001.

Finally, we considered more carefully how quickly the system converges to d^* for various values of n . There are approximately k^n possible states, so the convergence time could in principle be quite large. However, we suspect that the Markov chain that arises here is *rapidly mixing*, which means that it will converge significantly faster (see [51] for more details about rapid mixing). We

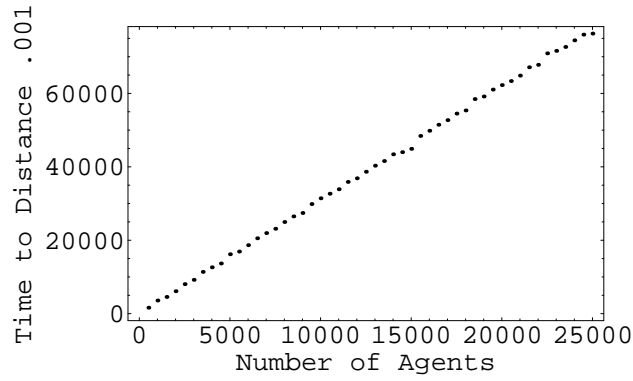


Figure 2.4: Average time to get within .001 of the minimum relative entropy distribution.

believe that the actually time needed is $O(n)$. This behavior is illustrated in Figure 2.4, which shows that for our example chain (again averaged over 10 runs), after approximately $3n$ steps, the Euclidean distance between the actual distribution of money in the system and d^* is less than .001. This suggests that we should expect the system to converge in a constant number of rounds per agent.

CHAPTER 3

IMPLICATIONS OF THE MODEL

3.1 Social Welfare and Scalability

In this section, we consider a fundamental question faced by system designers: what is the optimal amount of money and how does it depend on the size of the system? We discuss how our theoretical results from Sections 2.2 and 2.3 show that in order to maximize social welfare, the optimal amount of money is some constant per agent. Thus, a system designer that wants to maximize social welfare should manage the average quantity of money appropriately. However, we also show that this must be done carefully. Specifically, we show that increasing the amount of money improves performance up to a certain point, after which the system experiences a monetary crash. Once the system crashes, the only equilibrium will be the trivial one where all agents play s_0 . Thus, optimizing the performance of the system involves discovering how much money the system can handle before it crashes.

In Section 2.1, we define the game using a tuple $G = (T, \vec{f}, h, m, n)$. Thus, our definition of a game uses the average amount of money m rather than the equally reasonable total amount of money mhn . The choice is motivated by our theoretical results. Theorem 2.2.1 shows that the long-term distribution of money d^* depends on the average amount of money, but is independent n , provided it is sufficiently large. Thus, since we normalize δ_t by the number of agents in computing utility, the optimal threshold policy for the MDP developed in Appendix A.2 is also independent of n . Theorems 2.3.1 and 2.3.2 show that such policies constitute an ε -Nash equilibrium. Thus, modulo a technical

issue regarding the rate of convergence of the Markov Chain towards its stationary distribution, to determine the optimal amount of money for a large system, it suffices to determine the optimal value of m , the average amount of money per agent.

We remark that, in practice, it may be easier for the designer to vary the price of fulfilling a request than to control the amount of money in the system. This produces the same effect. For example, changing the cost of fulfilling a request from \$1 to \$2 is equivalent to halving the amount of money that each agent has. Similarly, halving the the cost of fulfilling a request is equivalent to doubling the amount of money that everyone has. With a fixed amount hmn of money, there is an optimal product hnc of the number hn of agents and the cost c of fulfilling a request.

This also tells us how to deal with a dynamic pool of agents. Our system can handle newcomers relatively easily: simply allow them to join with no money. This gives existing agents no incentive to leave and rejoin as newcomers. (By way of contrast, in systems where each new agent starts off with a small amount of money, such an incentive clearly exists.) We then change the price of fulfilling a request so that the optimal ratio is maintained. This method has the nice feature that it can be implemented in a distributed fashion; if all nodes in the system have a good estimate of n , then they can all adjust prices automatically. (Alternatively, the number of agents in the system can be posted in a public place.) Approaches that rely on adjusting the amount of money may require expensive system-wide computations (see [75] for an example), and must be carefully tuned to avoid creating incentives for agents to manipulate the system by which this is done.

Note that, in principle, the realization that the cost of fulfilling a request can change can affect an agent's strategy. For example, if an agent expects the cost to increase, then he may want to defer volunteering to fulfill a request. However, if the number of agents in the system is always increasing, then the cost always decreases, so there is never any advantage in waiting. There may be an advantage in delaying a request, but it is far more costly, in terms of waiting costs than in providing service, since we assume the need for a service is often subject to real waiting costs. In particular, many service requests, such as those for information or computation, cannot be delayed without losing most of their value.

Issues of implementation aside, we have now reduced the problem of determining the optimal total amount of money for a large system to that of determining the optimal average amount of money, independent of the exact number of agents. Before we can determine the optimal value of m , we have to answer a more fundamental question: given an equilibrium that arose for some value of m , how good is it?

Consider a single round of the game with a population of a single type t and an equilibrium threshold k . If a request is satisfied, social welfare increases by $\gamma_t - \alpha_t$; the requester gains γ_t utility and the satisfier pays a cost of α_t . If no request is satisfied then no utility is gained. What is the probability that a request will be satisfied? This requires two events to occur. First, the agent chosen to make a request must have a dollar, which happens with probability approximately $1 - \zeta$, where $\zeta = d^*(t, 0)$ is the fraction of agents with no money. Second, there must be a volunteer able and willing to satisfy the request. Any agent who does not have his threshold amount of money is willing to volunteer. Thus, if

$\theta = d^*(t, k_t)$ is the fraction of agents at their threshold, then the probability of having a volunteer is $1 - (1 - \beta_t)^{(1-\theta)n}$. We believe that in most large systems this probability is quite close to 1; otherwise, either β_t must be unrealistically small or θ must be very close to 1. For example, even if $\beta = .01$ (i.e., an agent can satisfy 1% of requests), 1000 agents will be able to satisfy 99.99% of requests. If θ is close to 1, then agents will have an easier time earning money than spending money (the probability of spending a dollar is at most $1/n$, while for large β the probability of earning a dollar if an agent volunteers is roughly $(1/n)(1/(1 - \theta))$). If an agent is playing s_4 and there are n rounds played a day, this means that for $\theta = .9$ he would be willing to pay α_t today to receive γ_t over 10 years from now. For most systems, it seems unreasonable to have δ_t or γ_t/α_t this large. Thus, for the purposes of our analysis, we approximate $1 - (1 - \beta_t)^{(1-\theta)n}$ by 1.

With this approximation, we can write the expected increase in social welfare each round as $(1 - \zeta)(\gamma_t - \alpha_t)$. Since $U_i(\vec{S})$ is normalized by the discount factor, the total expected social welfare summed over all rounds is also $(1 - \zeta)(\gamma_t - \alpha_t)$. If we have more than one type of agent, the situation is essentially the same. The equation for social welfare is more complicated because now the gain in welfare depends on the γ , α , and δ of the agents chosen in that round, but the overall analysis is the same, albeit with more cases. In the general case,

$$\zeta = \sum_t d^*(t, 0) \quad (3.1)$$

Thus our goal is clear: find the amount of money that, in equilibrium, minimizes ζ .

In general, as the following theorem shows, ζ decreases as m increases. More specifically, given our assumption that the system is starting at the greatest equilibrium \vec{k} , increasing m and then following best response dynamics leads to the

new greatest equilibrium \vec{k}' . As long as \vec{k}' is non-trivial, $\zeta_{m',\vec{k}'} \leq \zeta_{m,\vec{k}}$.

Theorems 2.2.1, 2.3.1, and 2.3.2 place requirements on the values of n and δ_t . Intuitively, the theorems require that the δ_t s is sufficiently large to ensure that agents are patient enough that their decisions are dominated by long-run behavior rather than short-term utility, and that n is sufficiently large to ensure that small changes in the distribution of money do not move it far from d^* . In the theorems in this section, assume that these conditions are satisfied. To simplify the statements of the theorems, we use “the standard conditions hold” to mean that the game $G = (T, \vec{f}, h, m, n)$ under consideration is such that $n > n^*$ and $\delta_t > \delta^*$ for the n^* and δ^* needed for the results of Theorems 2.2.1, 2.3.1, and 2.3.2 to apply.

Theorem 3.1.1. *Let $G = (T, \vec{f}, h, m, n)$ be such that the standard conditions hold, and let \vec{k} be the greatest equilibrium for G . Then if $m' > m$, the best-reply dynamics in $G' = (T, \vec{f}, h, m', n)$ starting at \vec{k} converge to some $\vec{k}' \leq \vec{k}$ that is the greatest equilibrium of G' . If \vec{k}' is a nontrivial equilibrium, then $\zeta_{m',\vec{k}'} \leq \zeta_{m,\vec{k}}$.*

Proof. Since \vec{k} is an equilibrium, $BR_G(\vec{k}) = \vec{k}$. By Lemma 2.3.2, BR_G is non-increasing in m . Thus, $\vec{k} = BR_G(\vec{k}) \geq BR_{G'}(\vec{k})$. Applying best-reply dynamics to $BR_{G'}$ starting at \vec{k} as in Theorem 2.3.2 gives us an equilibrium \vec{k}' such that $\vec{k}' \leq \vec{k}$. By Lemma 2.3.2, $BR_G(\vec{k}')$ is non-decreasing in \vec{k}' , so this is the greatest equilibrium. Suppose \vec{k}' is nontrivial. By Equations (2.2) and (3.1),

$$\zeta_{m,\vec{k}} = \sum_t d^*(t, 0) = \sum_t \frac{f_t q(t, 0)}{\sum_{j=0}^{k_t} \lambda_{m,\vec{k}}^j q(t, j)}.$$

Again by Lemma 2.3.2, $\lambda_{m,\vec{k}}$ is non-decreasing in m and non-increasing in \vec{k} . Thus, $\zeta_{m',\vec{k}'} \leq \zeta_{m,\vec{k}}$. □

Theorem 3.1.1 tells us that, as long as the system does not crash, more money is better. The following corollary tells us that such a crash is an essential feature; a sufficient increase in the amount of money leads to a monetary crash. Moreover, once the system has crashed, adding more money does not cause the system to become “uncrashed.”

Corollary 3.1.1. *Consider the family of games $G_m = (T, \vec{f}, h, m, n)$ such that the standard conditions hold. There exists a critical average amount m^* of money such that if $m < m^*$, then G_m has a nontrivial equilibrium, while if $m > m^*$, then G_m has no nontrivial equilibrium. (A nontrivial equilibrium may or may not exist if $m = m^*$.)*

Proof. To see that there is some m for which G_m has no nontrivial equilibrium, fix m . If there is no nontrivial equilibrium in G_m , we are done. Otherwise, suppose that the greatest equilibrium in G_m is \vec{k} . Choose $m' > \sum_t f_t k_t$, and let \vec{k}' be the greatest equilibrium in $G_{m'}$. By Theorem 3.1.1, $\vec{k}' \leq \vec{k}$. But if \vec{k}' is a nontrivial equilibrium then, in equilibrium, each agent of type t has at most $k'_t \leq k_t$ dollars. But then $m' > \sum_t f_t k_t \geq \sum f_t k'_t$, a contradiction.

Let m^* be the infimum over all m for which no nontrivial equilibrium exists in the game G_m . Clearly, by choice of m^* , if $m < m^*$, there is a nontrivial equilibrium in G_m . Now suppose that $m > m^*$. By the construction of m^* , there exists m' with $m > m' \geq m^*$ such that no nontrivial equilibrium exists in $G_{m'}$. Let the greatest equilibria with m' and m be \vec{k}' and \vec{k} , respectively. By Theorem 3.1.1, $\vec{k} \leq \vec{k}'$. Thus \vec{k} is also trivial. \square

Figure 3.1 shows an example of the monetary crash in the game

$$((.05, 1, 1, .95, 1), (.15, 1, 1, .95, 1)), (.3, .7), 10, m, 100).$$

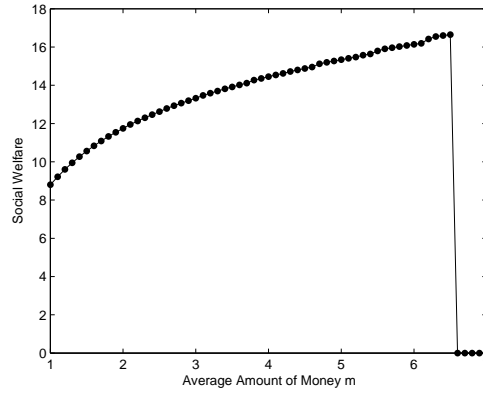


Figure 3.1: Social welfare for various average amounts of money, demonstrating a monetary crash.

In light of Corollary 3.1.1, the system designer should try to find m^* , the point where social welfare is maximized. We discuss how she might go about finding m^* in practice in Section 3.3. The system designer may wish to choose the m somewhat less than m^* . Since there will be a crash for any $m > m^*$, small changes in the characteristics of the population or mistakes by the designer in modeling them could lead to a crash if she chooses m too close to m^* .

The phenomenon of a monetary crash is intimately tied to our assumption of fixed prices. We saw such a crash in practice in the babysitting co-op example. If the price is allowed to float freely, we expect that, as the money supply increases, there will be inflation; the price will increase so as to avoid a crash. However, floating prices can create other monetary problems, such as speculation, booms, and busts. Floating prices also impose transaction costs on agents. In systems where prices would normally be relatively stable, these transaction costs may well outweigh the benefits of floating prices, so a system designer may opt for fixed prices, despite the risk of a crash.

3.2 Dealing with Nonstandard Agents

The model in Section 2.1 defines the utility of standard agents, who value service and dislike using their resources to provide it to others. This seems like a natural description of the way most people use distributed systems. However, in a real system, not every user will behave the way the designer intends. A practical system needs to be robust to nonstandard behaviors. In this section, we show how our model can be used to understand the effects of four interesting types of nonstandard behavior. First, an agent might provide service even when he will receive nothing in return, behaving as an altruist. Second, rather than viewing money as a means to satisfy future requests, an agent might place an inherent value on it and start hoarding it. Third, an agent might create additional identities, known as *sybils*, to try and manipulate the system. Finally, agents might collude with each other.

The results of this section give a system designer insight into how to design a script system that takes into account (and is robust to) a number of frequently-observed behaviors.

3.2.1 Altruists

P2P filesharing systems often have large numbers of free riders; they work because a small number of altruistic users provide most of the files. For example, Adar and Huberman [2] found that, in the Gnutella network, nearly 50 percent of responses are from the top 1 percent of sharing hosts. A wide variety of systems have been proposed to discourage free riding (see Section 1.2). However,

in our model, unless this system mostly eliminates the altruistic users, adding such a system will have no effect on rational users.

To make this precise, take an altruist to be someone who always volunteers to fulfill requests, regardless of whether the other agent can pay. Agent i might rationally behave altruistically if, rather than suffering a loss of utility when satisfying a request, i derives positive utility from satisfying it. Such a utility function is a reasonable representation of the pleasure that some people get from the sense that they provide the music that everyone is playing. For such altruistic agents, the strategy of always volunteering is dominant. While having a nonstandard utility function might be one reason that a rational agent might use this strategy, there are certainly others. For example a naive user of filesharing software with a good connection might well follow this strategy. All that matters for the following discussion is that there are some agents that use this strategy, for whatever reason. For simplicity, we assume that all such agents have the same type t_a .

Suppose that a system has a altruists. Intuitively, if a is moderately large, they will manage to satisfy most of the requests in the system even if other agents do no work. Thus, there is little incentive for any other agent to volunteer, because he is already getting full advantage of participating in the system. Based on this intuition, it is a relatively straightforward calculation to determine a value of a that depends only on the types, but not the number n , of agents in the system, such that the dominant strategy for all standard agents i is to never volunteer to satisfy any requests.

Proposition 3.2.1. *For all games $(T, \vec{f}, h, m, 1)$ with $f_{t_a} > 0$, there exists a value a such that, if $n > a/(f_{t_a} h)$, then never volunteering is a dominant strategy for all standard*

agents.

Proof. Consider the strategy for a standard agent i in the presence of a altruists. Even with no money, agent i will get a request satisfied with probability $1 - (1 - \beta_{t_a})^a$ just through the actions of the altruists. Consider a round when agent i is chosen to make a request. If he has no money (because he never volunteered), his expected utility is $\gamma_{\tau(i)}(1 - (1 - \beta_{t_a})^a)$. His maximum possible utility for the round is $\gamma_{\tau(i)}$. Thus, a strategy where he volunteers can increase his utility for a round by at most $\gamma_{\tau(i)}(1 - \beta_{t_a})^a$. Thus, even if the agent gets every request satisfied, his expected utility can increase by at most

$$\begin{aligned} & (1 - \delta_{\tau(i)}) \sum_{r=1}^{\infty} (\rho_{\tau(i)}/hn) \gamma_{\tau(i)} (1 - \beta_{t_a})^a (1 - (1 - \delta_{\tau(i)})/n)^r \\ = & (1 - \delta_{\tau(i)}) (\rho_{\tau(i)}/h) \gamma_{\tau(i)} (1 - \beta_{t_a})^a (1 - \delta_{\tau(i)}) \\ = & (\rho_{\tau(i)}/h) \gamma_{\tau(i)} (1 - \beta_{t_a})^a. \end{aligned}$$

Clearly this expression goes to 0 as a goes to infinity. If we take a large enough that the expression is less than α_t for all types t , then the value of having every future request satisfied is less than the cost of volunteering now, so no agent will ever volunteer. \square

Consider the following reasonable values for our parameters: $\beta_t = .01$ (so that each player can satisfy 1% of the requests), $\gamma_t = 1$, $\alpha_t = .1$ (a low but non-negligible cost), $\delta_t = .9999/\text{day}$ (which corresponds to a yearly discount factor of approximately 0.95), and an average of 1 request per day per player. Then as long as $a > 1145$ to ensure that not volunteering is a dominant strategy. While this is a large number, it is small relative to the size of a large P2P network.

Proposition 3.2.1 shows that with enough altruists, the system eventually experiences a monetary crash, since all agents will use a threshold of zero. How-

ever, interesting behavior can still arise with smaller numbers of altruists. Consider the situation where an a fraction of requests are immediately satisfied at no cost without the requester needing to ask for volunteers. Intuitively, these are the requests satisfied by the altruists, although the following result also applies to any setting where agents occasionally have a (free) outside option. The following theorem shows that social welfare is increasing in a .

Let $G = (\{t\}, 1, h, m, n)$ be a game with a single type for which the standard conditions hold. Consider the family G_a of games (parameterized by a) that result from G if a fraction a of requests can be satisfied at no cost. That is, the game G_a is the same as G , except that if an agent i makes a request, with probability a , it is satisfied at no cost, and with probability $1 - a$, an agent is chosen among the volunteers to satisfy the request, just as in G , and the i is charged 1 dollar to have the request satisfied.

Theorem 3.2.1. *For the interval of values of a where there is no monetary crash in G_a , social welfare increases as a increases (assuming that the greatest equilibrium is played by all agents in G_a).*

Proof. An agent's utility in a round where he makes a request and it is satisfied at no cost is γ_t . Since such rounds occur with probability a , by assumption, our normalization guarantees that the sum of an agent's expected utility in rounds where a request is satisfied at no cost is $a\gamma_t$. The same analysis as in Section 3.1 shows that the some of an agent's expected utility in the remaining rounds is $(1 - a)(1 - \zeta(a))(\gamma_t - \alpha_t)$, where, as before, $\zeta(a) = d^*(t, 0, a)$, the equilibrium value of $d^*(t, 0)$ in the game G_a . Thus, expected utility as a function of a is

$$a\gamma_t + (1 - a)(1 - \zeta(a))(\gamma_t - \alpha_t). \quad (3.2)$$

To see that this expression increases as a increases, we would like to take the derivative relative to a and show it is positive. Unfortunately, $\zeta(a)$ may not even be continuous. Because strategies are integers, there will be regions where $\zeta(a)$ is constant, and then a jump when a critical value of a is reached that causes the equilibrium to change. At a point a in a region where $\zeta(a)$ is constant, $\zeta'(a) = 0$, so the derivative of Equation (3.2) is $\gamma_t - (1 - \zeta(a))(\gamma_t - \alpha_t) > 0$. Hence, social welfare is increasing at such points.

Now consider a point a where $\zeta(a)$ is discontinuous. Such a discontinuity occurs when the greatest equilibrium, the greatest value \vec{k} for which $BR_{G_a}(\vec{k}) = \vec{k}$, changes. We show that, for a fixed \vec{k} , $BR_{G_a}(\vec{k})$ is non-increasing in a . Since increasing a can only cause the $BR_{G_a}(\vec{k})$ to decrease, the discontinuity must be caused by a change from an equilibrium \vec{k} to a new equilibrium $\vec{k}' < \vec{k}$. Fix a vector \vec{k} of thresholds, and let $p_u^{\vec{k},m,a}$ be the probability that i will be earn a dollar in a given round if he is willing to volunteer, given that a fraction a of requests is satisfied at no cost (so that $p_u^{\vec{k},m,0}$ is what we earlier called $p_u^{\vec{k},m}$); we similarly define $p_d^{\vec{k},m,a}$. It is easy to see that $p_u^{\vec{k},m,a} = (1 - a)p_u^{\vec{k},m,0}$ and $p_d^{\vec{k},m,a} = (1 - a)p_d^{\vec{k},m,0}$. The random variable $J(\kappa, p_u, p_d)$ in Equation (2.4) describes the first time at which an agent starting with κ dollars and using the threshold κ while earning a dollar with probability p_u and spending a dollar with probability p_d reaches zero dollars. As a increases, $p_u^{\vec{k},m,a}$ and $p_d^{\vec{k},m,a}$ both decrease, but the ratio $p_u^{\vec{k},m,a}/p_d^{\vec{k},m,a}$ remains constant. Intuitively, this means that the agent “slows down” his random walk on amounts of money by a factor of $1/(1 - a)$. Thus, the value of the expectation in Equation (2.4), and hence the right-hand side of Equation (2.4), is decreasing as a function of a . By Lemma 2.3.1, $(BR_{G_a}(\vec{k}))_t$ is the maximum value of κ such that Equation (2.4) is satisfied. Decreasing the right-hand side can only decrease the maximum value of κ , so $BR_{G_a}(\vec{k})$ is non-increasing as a function of

a.

By Lemma 2.3.2, $\lambda_{m,\vec{k}}$ is non-increasing in \vec{k} (unless the system crashes). Since, as we have just shown, if there is a discontinuity at $\zeta(a)$ when a increases, the greatest equilibrium changes at a from \vec{k} to $\vec{k}' < \vec{k}$, we must have $\lambda_{m,\vec{k}'} \geq \lambda_{m,\vec{k}}$. In Equation (2.2) for $i = 0$, the value of the numerator is independent of λ , but the denominator with $\lambda_{m,\vec{k}'}$ is greater than or equal to the denominator with $\lambda_{m,\vec{k}}$. Thus $d^*(t, 0, a) = \zeta(a)$ is non-increasing at a . By Equation (3.2), this means that expected utility is increasing at a . Thus, in either case, social welfare is increasing in a . \square

Theorem 3.2.1 and Proposition 3.2.1 combine to tell us that a little altruism is good for the system, but too much causes a crash. Figure 3.2 demonstrates this phenomenon. As we saw in Section 3.1, such crashes are caused when m , the average amount of money, is too large. By decreasing m appropriately, even relatively large values of a can be exploited, as Figure 3.3 shows. The “social welfare without adjustment” plot is the same data from Figure 3.2, with the corresponding plot of the amount of money horizontal since m was held fixed. By decreasing the average amount of money appropriately as the number of altruists increases, a system designer can increase social welfare while avoiding a crash.

3.2.2 Hoarders

Whenever a system allows agents to accumulate something, be it work done, as in SETI@home, friends on online social networking sites, or “gold” in an on-line game, a certain group of users seems to make it their goal to accumulate

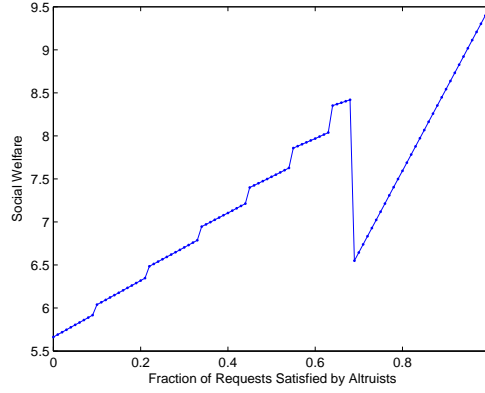


Figure 3.2: Altruists can cause a crash.

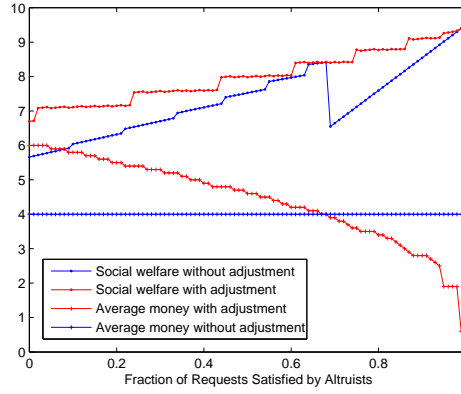


Figure 3.3: m can be adjusted as a increases.

as much of it as possible. In pursuit of this, they will engage in behavior that seems irrational. For simplicity here, we model hoarders as playing the strategy s_∞ . This means that they will volunteer under all circumstances. Our analysis would not change significantly if we also required that they never made a request for work. Our first result shows that, for a fixed money supply, having more hoarders makes standard agents worse off.

Consider a game $G = (T, \vec{f}, h, m, n)$ such that the standard conditions hold. Consider the family G_{f_h} of games (parameterized by f_h) that result from G if a

fraction f_h of agents are hoarders. That is, $G_{f_h} = (T \times \{0, 1\}, \vec{f}', h', m, n)$ where at agent of type $(t, 0)$ is a standard agent of type t , but an agent of type $(t, 1)$ is a hoarder and always uses the strategy s_∞ (his probabilities are still determined by β_t , ρ_t , and χ_t). Define \vec{f}' by taking $f'_{(t,0)} = (1 - f_h)f_t$ and $f'_{(t,1)} = f_h f_t$ for all types t . Let h' be the smallest multiple of h such that $f_{(t,i)}h'$ is an integer for all t and i . (We need to adjust h because otherwise the number of agents in the base game may not be well defined.) Finally, to account for the changed h , let $\delta_{(t,i)} = 1 - (1 - \delta_t)h/h'$.

Theorem 3.2.2. *In the family G_{f_h} of games, social welfare is non-increasing in f_h (if the greatest equilibrium is played by all agents in G_{f_h}).*

Proof. Let $\vec{k}(f_h)$ denote the greatest equilibrium in G_{f_h} . An increase in f_h is equivalent to taking some number of standard agents and increasing their strategy to s_∞ . It follows from Lemma 2.3.2 that $BR_{G_{f_h}}$ is non-decreasing in f_h , and so $\vec{k}(f_h)$ is non-decreasing in f_h . Again by Lemma 2.3.2, $\lambda_{m, \vec{k}(f_h)}$ is non-increasing in f_h . Let $\zeta f_h = 1/(1 - f_h) \sum_t d^*((t, 0), 0, f_h)$ be the fraction of non-hoarders with zero dollars, where $d^*((t, 0), 0, f_h)$ is the value of $d^*((t, 0), 0)$ at the greatest equilibrium of G_{f_h} . By Equation (2.2), $\zeta(f_h)$ is non-decreasing in f_h . Thus, social welfare is non-increasing in f_h . \square

Hoarders do have a beneficial aspect. As we have observed, a monetary crash occurs when a dollar becomes valueless, because there are no agents willing to take it. However, with hoarders in the system, there is always someone who will volunteer, so there cannot be a crash. Thus, for any m , the greatest equilibrium will be nontrivial and, by Theorem 3.1.1, social welfare keeps increasing as m increases. So, in contrast to altruism, where the appropriate response was to decrease m , the appropriate response to hoarders is to increase m . In fact, our

results indicate that the optimal response to hoarders is to make m infinite. This is due to our unrealistic assumption that hoarders would use the strategy s_∞ regardless of the value of m . There is likely an upper limit on the value of m in practice, since it is unlikely that hoarders would be willing to hoard scrip if it is so easily available.

3.2.3 Sybils

Unless identities in a system are tied to a real world identity (for example by a credit card), it is effectively impossible to prevent a single agent from having multiple identities [22]. Nevertheless, there are a number of techniques that can make it relatively costly for an agent to do so. For example, Credence uses cryptographic puzzles to impose a cost each time a new identity wishes to join the system [76]. Given that a designer can impose moderate costs to sybiling, how much more need she worry about the problem? In this section, we show that the gains from creating sybils when others do not diminish rapidly, so modest costs may well be sufficient to deter sybiling by typical users. However, sybiling is a self-reinforcing phenomenon. As the number of agents with sybils gets larger, the cost to being a non-sybiling agent increases, so the incentive to create sybils becomes stronger. Therefore, measures to discourage or prevent sybiling should be taken early before this reinforcing trend can start. Finally, we examine the behavior of systems where only a small fraction of agents have sybils. We show that under these circumstances a wide variety of outcomes are possible (even when all agents are of a single type), ranging from a crash (where no service is provided) to an increase in social welfare. This analysis provides insight into the tradeoffs between efficiency and stability that occur when con-

trolling the money supply of the system's economy.

When an agent of type t creates sybils, the only parameter of his type that may change as a result is χ_t , if we redefine the likelihood of an agent being chosen to be the likelihood of the agent or any of his sybils being chosen. For simplicity, we assume that each sybil is as likely to be chosen as the original agent, so creating s sybils increases χ_t by $s\chi_t$. (Sybils may have other impacts on the system, such as increased search costs, but we expect these to be minor.)

Increasing χ_t benefits an agent by increasing his value of ω_t and thus p_u , his probability of earning a dollar (see Equation (A.2) in Appendix A.2). When $p_u < p_d$, the agent has more opportunities to spend money than to earn money, so he will regularly have requests go unsatisfied due to a lack of money. In this case, the fraction of requests he has satisfied is roughly p_u/p_d , so increasing p_u by creating sybils results in a roughly linear increase in utility. As Theorem 3.2.3 shows, when p_u is close to p_d , the increase in satisfied requests is no longer linear, so the benefit of increasing p_u begins to diminish. Finally, when $p_u > p_d$, most of the agent's requests are being satisfied, so the benefit from increasing p_u is very small. Figure 3.4 illustrates an agent's utility as p_u varies for $p_d = .0001$.¹ We formalize the relationship between p_u , p_d , and the agent's utility in the following theorem, whose proof is deferred to Appendix A.3.

Theorem 3.2.3. *Fix a game G and vector of thresholds \vec{k} . Let $R_{\vec{k},t} = p_u^{\vec{k},t}/p_d^t$. In the limit as the number of rounds goes to infinity, the fraction of the agent's requests that have an agent willing and able to satisfy them that get satisfied is $(R_{\vec{k},t} - R_{\vec{k},t}^{k_t+1})/(1 - R_{\vec{k},t}^{k_t+1})$ if $R_{\vec{k},t} \neq 1$ and $k_t/(k_t + 1)$ if $R_{\vec{k},t} = 1$.*

¹Except where otherwise noted, the remaining figures in this section assume that $m = 4$, $n = 10000$ and that there is a single type of rational agent with $\alpha = .08$, $\beta = .01$, $\gamma = 1$, $\delta = .97$, $\rho = 1$, and $\chi = 1$. These values are chosen solely for illustration, and are representative of a broad range of parameter values. The figures are based on calculations of the equilibrium behavior.

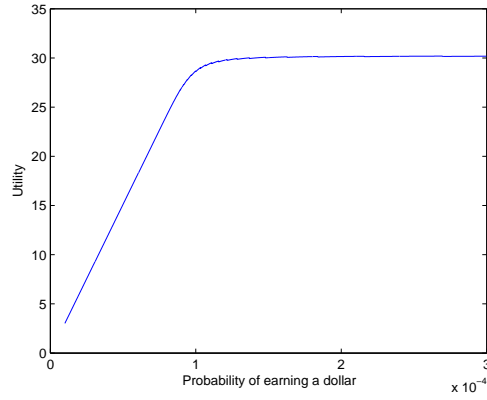


Figure 3.4: The effect of p_u on utility

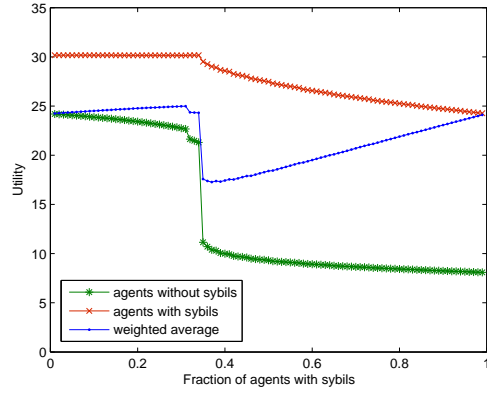


Figure 3.5: The effect of sybils on utility

Theorem 3.2.3 gives insight into the equilibrium behavior with sybils. Clearly, if sybils have no cost, then creating as many as possible is a dominant strategy. However, in practice, we expect there is some modest overhead involved in creating and maintaining a sybil, and that a designer can take steps to increase this cost without unduly burdening agents. With such a cost, adding a sybil might be valuable if p_u is much less than p_d , and a net loss otherwise. This makes sybils a self-reinforcing phenomenon. When a large number of agents create sybils, agents with no sybils have their p_u significantly decreased. This makes them much worse off and makes sybils much more attractive to them.

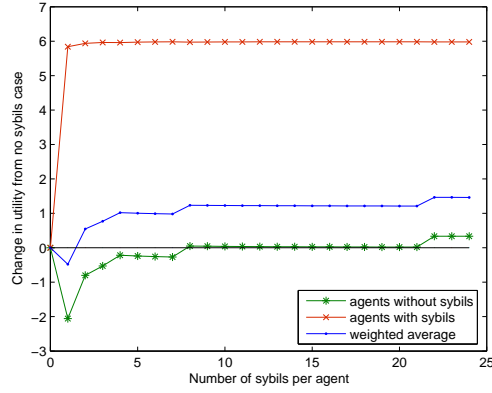


Figure 3.6: Sybils can improve utility

Figure 3.5 shows an example of this effect. This self-reinforcing quality means that it is important to take steps to discourage the use of sybils before they become a problem. Luckily, Theorem 3.2.3 also suggests that a modest cost to create sybils will often be enough to prevent agents from creating them because with a well chosen value of m , few agents should have low values of p_u .

We have interpreted Figures 3.4 and 3.5 as being about changes in χ due to sybils, but the results hold regardless of what caused differences in χ . For example, agents may choose a volunteer based on characteristics such as connection speed or latency. If these characteristics are difficult to verify and do impact decisions, our results show that agents have a strong incentive to lie about them. This also suggests that the decision about what sort of information the system should enable agents to share involves tradeoffs. If advertising legitimately allows agents to find better service or more services they may be interested in, then advertising can increase social welfare. But if these characteristics impact decisions but have little impact on the actual service, then allowing agents to advertise them can lead to a situation like that in Figure 3.5, where some agents have a significantly worse experience.

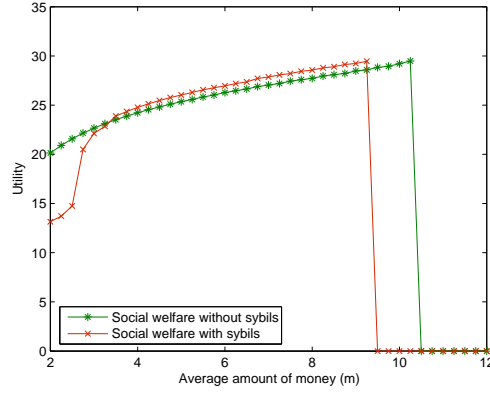


Figure 3.7: Sybils can cause a crash

We have seen that when a large fraction of agents have sybils, those agents without sybils tend to be starved of opportunities to work (i.e. they have a low value of p_u). However, as Figure 3.5 shows, when a small fraction of agents have sybils this effect (and its corresponding cost) is small. Surprisingly, if there are few agents with sybils, an increase in the number of sybils these agents have can actually result in a *decrease* of their effect on the other agents. Because agents with sybils are more likely to be chosen to satisfy any particular request, they are able to use lower thresholds and reach those thresholds faster than they would without sybils, so fewer are competing to satisfy any given request. Furthermore, since agents with sybils can almost always pay to make a request, they can provide more opportunities for other agents to satisfy requests and earn money. Social welfare is essentially proportional to the number of satisfied requests (and is exactly proportional to it if everyone shares the same values of α and γ), so a small number of agents with a large number of sybils can improve social welfare, as Figure 3.6 shows. Note that, although social welfare increases, some agents may be worse off. For example, for the choice of parameters in this example, social welfare increases when twenty percent of agents create at least

two sybils, but agents without sybils are worse off unless the twenty percent of agents with sybils create at least eight sybils. As the number of agents with sybils increases, they start competing with each other for opportunities to earn money and so adopt higher thresholds, and this benefit disappears. This is what causes the discontinuity in Figure 3.5 when approximately a third of the agents have sybils.

This observation about the discontinuity also suggests another way to mitigate the negative effects of sybils: increase the amount of money in the system. This effect can be seen in Figure 3.7, where for $m = 2$ social welfare is very low with sybils but by $m = 4$ it is higher than it would be without sybils.

Unfortunately, increasing the average amount of money has its own problems. Recall from Section 3.1 that, if the average amount of money per agent is too high, the system will crash. It turns out that just a small number of agents creating sybils can have the same effect, as Figure 3.7 shows. With no sybils, the point at which social welfare stops increasing and the system crashes is between $m = 10.25$ and $m = 10.5$. If one-fifth of the agents each create a single sybil, the system crashes if $m = 9.5$, a point where, without sybils, the social welfare was near optimal. Thus, if the system designer tries to induce optimal behavior without taking sybils into account, the system will crash. Moreover, because of the possibility of a crash, raising m to tolerate more sybils is effective only if m was already set conservatively.

This discussion shows that the presence of sybils can have a significant impact on the tradeoff between efficiency and stability. Setting the money supply high can increase social welfare, but at the price of making the system less stable. Moreover, as the following theorem shows, whatever efficiencies can be

achieved with sybils can be achieved without them, at least if there is only one type of agent. In the theorem, we consider a system where all agents have the same type t . Suppose that some subset of the agents have created sybils, and all the agents in the subset have created the same number of sybils. We can model this by simply taking the agents in the subsets to have a new type s , which is identical to t except that the value of χ increases. Thus, we state our results in terms of systems with two types of agents, t and s .

Theorem 3.2.4. *Suppose that t and s are two types that agree except for the value of χ , and that $\chi_t < \chi_s$. If $\vec{k} = (k_t, k_s)$ is an ε -Nash equilibrium for $G = (\{t, s\}, \vec{f}, h, m, n)$ with social welfare w , then there exist h' , m' , and n' such that $\vec{k}' = (k_s)$ is an ε -Nash equilibrium for $G'_{h', m', n'} = (\{t\}, \{1\}, h', m', n')$ with social welfare greater than w .*

We defer proof of Theorem 3.2.4 to Appendix A.3.

The analogous result for systems with more than one type of agent is not true. Figure 3.5 shows a game with a single type of agent, some of whom have created two sybils. However, we can reinterpret it as a game with two types of agents, one of whom has a larger value of χ . With this reinterpretation, Figure 3.5 shows that social welfare is higher when all the agents are of the type t_h with the higher value of χ than when only 40% are. Moreover, if only 40% of the agents have type t_h , social welfare would increase if the remaining agents created two sybils each (resulting in all agents having the higher value of χ). Note that this situation, where there are two types of agents, of which one has a higher value of χ , is exactly the situation considered by Theorem 3.2.4. Thus, the theorem shows that for any equilibrium with two such types of agents, there is a better equilibrium where one of those types creates sybils so as to effectively create only one type of agent.

While situations like this show that it is theoretically possible for sybils to increase social welfare beyond what is possible to achieve by simply adjusting the average amount of money, this outcome seems unlikely in practice. It relies on agents creating just the right number of sybils. For situations where such a precise use of sybils would lead to a significant increase in social welfare, a designer could instead improve social welfare by biasing the algorithm agents use for selecting which volunteer will satisfy the request.

Thus far, we have assumed that when agents create sybils the amount of money in the system does not change. However, the presence of sybils increases the number of apparent agents in the system. Since social welfare depends on the average amount of money per agent, if the system designer mistakes these sybils for an influx of new users and increases the money supply accordingly, she will actually end up increasing the average amount of money in the system, and may cause a crash. This emphasizes the need for continual monitoring of the system rather than just using simple heuristics to set the average amount of money, an issue we discuss more in Section 3.3.

3.2.4 Collusion

Agents that collude gain two major benefits. The primary benefit is that they can share money, which makes them less likely to run out of money (and hence unable to make a request), and allows them to pursue a joint strategy for determining when to work. A secondary benefit, but important in particular for larger collusive groups, is that they can satisfy each other's requests. The effects of collusion on the rest of the system depend crucially on whether agents are

able to volunteer to satisfy requests when they personally cannot satisfy the request but one of their colluding partners can. In a system where a request is for computation, it seems relatively straightforward for an agent to pass the computation to a partner to perform and then pass the answer back to the requester. On the other hand, if a request is a piece of a file it seems less plausible that an agent would accept a download from an unexpected source, and it seems wasteful to have the chosen volunteer download it for the sole purpose of immediately uploading it. If it is possible for colluders to pass off requests in this fashion, they are able to effectively act as sybils for each other, with all the consequences discussed in Section 3.2.3. However, if agents can volunteer only for requests they can personally satisfy, the effects of collusion are almost entirely positive.

Since we have already discussed the consequences of sybils, we will assume that agents are able to volunteer only to satisfy requests that they personally can satisfy. Furthermore, we make the simplifying assumption that agents that collude are of the same type, because if agents of different types collude their strategic decisions become more complicated. For example, once the colluding group has accumulated a certain amount of money, it may wish to have only members with small values of α volunteer to satisfy requests; or when it is low on money, it may wish to deny use of money to members with low values of γ . This results in strategies that involve sets of thresholds rather than a single threshold. While there seems to be nothing fundamentally different about the situation, it makes calculations significantly more difficult.

With these assumptions, we now examine how colluding agents will behave. Because colluding agents share money and types, it is irrelevant which members

actually perform work and have money. All that matters is the total amount of money the group has. This means that when the group needs money, everyone in the group volunteers for a job; otherwise, no one does. Thus, the group essentially acts like a single agent, using a threshold that is somewhat less than the sum of the thresholds that the individual agents would have used, because it is less likely that c agents will make ck requests in rapid succession than a single agent making k . Furthermore, some requests will not require scrip at all because they can potentially be satisfied by other members of the colluding group. When deciding whether the group should satisfy a member's request or ask for an outside volunteer to fulfill it, the group must decide whether it should pay a cost of α to avoid spending a dollar. Since not spending a dollar is effectively the same as earning a dollar, the decision is already optimized by the threshold strategy; the group should always attempt to satisfy a request internally unless it is in a temporary situation where the group is above its threshold.

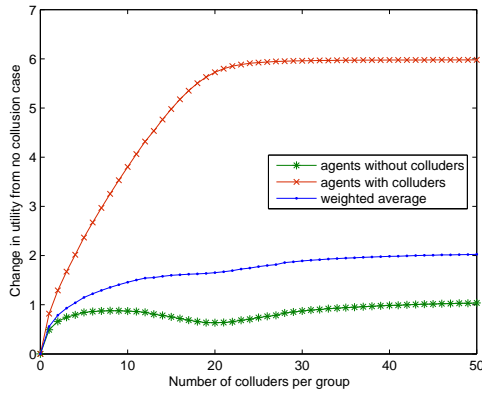


Figure 3.8: The effect of collusion on utility

Figure 3.8 shows an example of the effects of collusion on agents' utilities as the size of collusive groups increases. As this figure suggests, the effects typically go through three phases. Initially, the fraction of requests colluders

satisfy for each other is small. This means that each collusive group must work for others to pay for almost every request its members make. However, since they share money, the colluders do not have to work as often as individuals would. Thus, other agents have more opportunity to work, and every agent's p_u increases, making all agents better off.

As the number of colluders increases, the fraction of requests they satisfy internally grows significant. We can think of p_d as decreasing in this case, and view these requests as being satisfied “outside” the scrip system because no scrip changes hands. This is good for colluders, but is bad for other agents whose p_u is lower, since fewer requests are being made. Even in this range, non-colluding agents still tend to be better off than if there were no colluders, because the overall competition for opportunities to work is still lower. Finally, once the collusive group is large enough, it will have a low p_d relative to p_u . This means the collusive group can use a very low threshold, which again begins improving utility for all agents. The analogous situation with sybils is transitory, and disappears when more agents create sybils. However, with collusion, this low threshold is an inherent consequence of colluders satisfying each other's requests, and so persists and even increases as the amount of collusion in the system increases. Since collusion is difficult to maintain (the problem of incentivizing agents to contribute is the whole point of using scrip), we would expect the size of collusive groups seen in practice to be relatively small. Therefore, we expect that for most systems collusion will make no agent worse off, and some better off. Note that, as with sybils, the decreased in competition that results from collusion can also lead to a crash. However, if the system designer is monitoring the system, and encouraging and expecting collusion, she can reduce m appropriately and prevent a crash.

These results also suggest that creating the ability to take out loans (with an appropriate interest rate) is likely to be beneficial. Loans gain the benefits of reduced competition without the accompanying cost of fewer requests being made in the system. However, implementing a loan mechanism requires addressing a number of other incentive problems. For example, *whitewashing*, where agents take on a new identity (in this case to escape debts) needs to be prevented [26].

3.3 Identifying User Strategies

In Section 2.2, we used relative entropy to derive an explicit formula for the distribution of money d^* given a game (T, \vec{f}, h, n, m) and vector of strategies \vec{k} . In this section, we want to go in the opposite direction: given the distribution of money, we want to infer the strategies \vec{k} , the set of types present T , and the fraction of each type \vec{f} . For those interested in studying the agents of a scrip system, knowing the fraction of agents using each strategy can provide a window into the preferences of those agents. For system designers, this knowledge is useful because, as we show in Section 3.1, how much money the system can handle without crashing depends on the fraction of agents of each type.

In equilibrium, the distribution of money has the form described in Lemma 2.2.1. Note that, in general, we do not expect to see exactly this distribution at any given time, but it follows from Theorem 2.2.1 that, after sufficient time, the distribution is unlikely to be very far from it. Does this distribution help us identify the strategies and types of agents?

As a first step to answering this question, given a distribution of money d

(where $d(i)$ is the fraction of agents with i dollars) such that $d(i)$ is rational for all i (this constraint is necessary of $d(i)$ is to represent the fraction of agents with i dollars in a real system), suppose that the maximum amount of money to which d gives positive probability is K_d . A vector \vec{f} of length $K_d + 1$ whose components are all rational numbers, where f_i is intuitively the fraction of agents playing the threshold strategy s_i , is an *explanation* of d if there exists a λ such that

$$d(j) = \sum_i d_\lambda(i, j),$$

where

$$d_\lambda(i, j) = f_i \lambda^j / \left(\sum_{l=0}^i \lambda^l \right) \quad (3.3)$$

if $j \leq i$ and 0 otherwise. Note that Equation (3.3) is very similar to Equation (2.2) from Lemma 2.2.1. In the following lemma, we show why we call \vec{f} an explanation: given a distribution d and an explanation \vec{f} we can find a game G where \vec{f} is the fraction of agents of each type and d is the equilibrium distribution of money (by which we mean that the value of d^* in Lemma 2.2.1 is such that $d(i) = \sum_t d^*(t, i)$).

Lemma 3.3.1. *If \vec{f} is an explanation for d , then there exists a game $G = (T, \vec{f}, h, m, n)$ and vector \vec{k} of thresholds such that \vec{k} is an ε -Nash equilibrium for G and the equilibrium distribution of money is d .*

Proof. Let $T = \{0, \dots, K_d\}$, h be the minimum integer such that $hd(i)$ is an integer for all i , $m = \sum_i id(i)$, and \vec{k} be such that $k_i = i$. For each type i , choose β_i , χ_i , and ρ_i arbitrarily, subject to the constraint that $\beta\chi/\rho = 1$ (so that, by definition, $\omega_i = 1$ for all types i). Finally, choose an arbitrary n .

By Lemma 2.3.1, for any n , an optimal threshold policy in the MDP $\mathcal{P}_{G, \vec{f}(\vec{k}), i}$

for an agent of type i is s_κ , where κ is the maximum value such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u, p_d)}] \gamma_t. \quad (2.4)$$

Fix δ_i and γ_i , and let $g(\kappa)$ be the sequence of values of the right hand side of Equation (2.4) for natural numbers κ . Recall that the random variable $J(\kappa, p_u, p_d)$ represents the round at which an agent starting with κ dollars runs out of money. Since $J(0, p_u, p_d) = 0$ for all histories, $g(0) = \gamma_i$. The time at which an agent runs out of money is increasing in his initial amount of money. Thus, $J(\kappa, p_u, p_d)$ is a strictly decreasing function of κ , so $g(\kappa)$ is also strictly decreasing. Choose α_i such that $g(i+1) < \alpha_i < g(i)$.

Thus, we have established parameters $(\alpha_i, \beta_i, \gamma_i, \delta_i, \chi_i, \rho_i)$ for each type i so that s_i an optimal policy for agents of type i in the MDP $\mathcal{P}_{G, \vec{s}(\vec{k}), i}$. By Theorem 2.3.1, taking n and the δ_i sufficiently large makes \vec{k} a ε -Nash equilibrium for G . By Lemma 2.2.1, the equilibrium distribution of money is d . \square

In general, there is not a unique explanation of a distribution d . Say that a distribution of money d is *fully-supported* if there do not exist i and j such that $i < j$, $d(j) > 0$, and $d(i) = 0$. For any game G , if all agents play threshold strategies then the resulting distribution will be fully-supported because it has the form given in Lemma 2.2.1. As the following lemma shows, a fully-supported distribution can be explained in an infinite number of different ways.

Lemma 3.3.2. *If d is a fully-supported distribution of money with finite support, there exist an infinite number of explanations of d .*

We defer the proof of Lemma 3.3.2 to Appendix A.4.

Lemma 3.3.2 shows that d has an infinite number of explanations.

Lemma 3.3.1 shows that we can find an (approximate) equilibrium corresponding to each of them. The explanations \vec{f} we construct in the proof of Lemma 3.3.2 seem unnatural; typically $f_i > 0$ for all i . We are interested in a more parsimonious explanation, one that has a small support (i.e., the number of thresholds i for which $f_i > 0$ is small), for reasons the following lemma makes clear.

Lemma 3.3.3. *Let \vec{f} be an explanation for d . If s is the size of the support of \vec{f} , then any other explanation will have a support of size at least $K_d - s$.*

Proof. Suppose that \vec{f} is an explanation for d . By Lemma 3.3.1, there is a game $G = (T, \vec{f}, h, m, n)$ and vector \vec{k} of thresholds such that \vec{k} is an ϵ -Nash equilibrium for G and the equilibrium distribution of money is d . Moreover, the proof of Lemma 3.3.1 shows that we can take $T = \{0, \dots, K_d\}$, $k_i = i$, and $\omega_i = 1$ for each type $i \in T$. By Equation (2.2) in Lemma 2.2.1, $d^*(t, i) = f_t \lambda^i q(t, i) / \sum_{j=0}^{k_t} \lambda^j q(t, j)$, where λ is the (unique) value that satisfies Equation (2.3). We first show that if $f_{i-1} = 0$, then $d(i)/d(i-1) = \lambda$. Since, for all i , $\omega_i = 1$, it is immediate from Equation (A.1) in the appendix that $q(i, j) = q(i, j')$ for all j and j' . Thus, the q terms cancel, so $d^*(i, j) = f_i \lambda^j / \sum_{l=0}^{k_i} \lambda^l$. Let $b_i = f_i / \sum_{l=0}^{k_i} \lambda^l$; then $d^*(i, j) = \lambda^j b_i$. Only agents with a threshold of at least j can have j dollars, so

$$d(j) = \sum_i d^*(i, j) = \sum_{\{t: l \geq j\}} d^*(l, j) = \sum_{\{t: l \geq j\}} b_l \lambda^j = B_j \lambda^j,$$

where $B_j = \sum_{\{t: l \geq j\}} b_l$. If $f_{i-1} = 0$, then $B_i = B_{i-1}$, so $d(i)/d(i-1) = \lambda$.

Since s strategies get positive probability according to \vec{f} , at least $K_d - s$ of the ratios $d(i)/d(i-1)$ with $1 \leq i \leq K_d$ must have value λ . Any other explanation \vec{f}' will have different coefficients f_i in Equation (3.3), so the value λ' satisfying it will also differ (since the requirement that $d(K_d) = d_\lambda(K_d, K_d)$ uniquely defines a value of λ). This means that the $K_d - s$ ratios with value λ must correspond to

strategies i such that $f_i > 0$. Thus, the support of any other explanation must be at least $K_d - s$. \square

If $s \ll K_d$, Lemma 3.3.3 gives us a strong reason for preferring the minimal explanation (i.e., the one with the smallest support); any other explanation will involve significantly more types of agents being present. For $s = 3$ and $K_d = 50$, the smallest explanation has a support of three thresholds, and thus requires three types; the next smallest explanation requires at least 47 types. Thus, if the number of types of agents is relatively small, the minimal explanation will be the correct one.

The proof of Lemma 3.3.3 also gives us an algorithm for finding this minimal explanation. Since $d(i) = B_i \lambda^i$, taking logs of both sides, $\log d(i) = \log B_i + i \log \lambda$. Because B_i is constant in ranges of i where $f_i = 0$, a plot of $\log d(i)$ will be a line with slope λ in these ranges. Thus, the minimal explanation can be found by finding the minimum number of lines of constant slope that fit the data. For a simple example of how such a distribution might look, Figure 3.9 shows an equilibrium distribution of money for the game

$$((.05, 1, 1, .95, 1), (.15, 1, 1, .95, 1)), (.3, .7), 10, 4, 100)$$

so the only difference between the types is that it costs the second type three times as much to satisfy a request) and the equilibrium strategy profile (20, 13). Figure 3.10 has the same distribution plotted on a log scale. Note the two lines with the same slope (λ) and the break at 13.

This procedure allows us to infer a distribution of money the minimal explanation of the number of types of agents: the fraction of the population composed of each type, and the strategy each type is playing. (Note that we cannot dis-

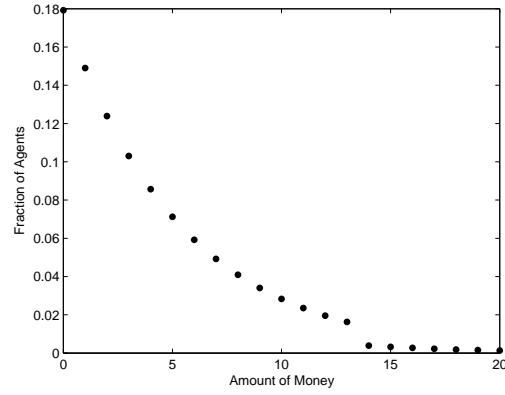


Figure 3.9: Distribution of money with two types of agents.

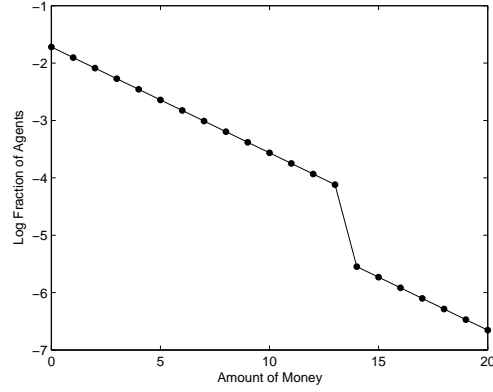


Figure 3.10: Log of the distribution of money with two types of agents.

tinguish multiple types with a shared ω_t playing the same strategy.) We would like to use this information to learn about the preferences of agents: their values of α_t , γ_t , and δ_t . Lemma 2.3.1 shows how we can do this. Once we find an explanation, the value of λ determines p_u^t and p_d^t for each type t . Then Equation 2.4 puts constraints on the values of α_t , β_t , and γ_t . Over time, if T , the set of types, remains constant, but \vec{f} , n , and m all vary as agents join and leave the system, a later observation with a slightly would give another equilibrium with new constraints on the types of the agents. A number of such observations potentially reveal enough information to allow strong inferences about agent types.

Thus far we have implicitly assumed that there are only a small number of types of agents in a system. Given that a type is defined by six real numbers, it is perhaps more reasonable to assume that each agent has a different type, but there is a small number of “clusters” of agents with similar types. For example, we might believe that generally agents either place a high value or a low value on receiving service. While the exact value may vary, the types of two low-value agents or two high-value agents will be quite similar. We have also assumed in our analysis that all agents play their optimal threshold strategy. However, computing this optimum may be too difficult for many agents. Even ignoring computational issues, agents may have insufficient information about their exact type or the exact types of other agents to compute the optimal threshold strategy. Both the assumption that there are a few clusters of agents with similar, but not identical, types and the assumption that agents do not necessarily play their optimal threshold strategy, but do play a strategy close to optimal, lead to a similar picture of a system, which is one that we expect to see in practice: we will get clusters of agents playing similar strategies (that is, strategies with thresholds clustered around one value), rather than all agents in a cluster playing exactly the same strategy. This change has relatively little impact on our results. Rather than seeing straight lines representing populations with a sharp gap between them, as in Figure 3.10, we expect slightly curved lines representing a cluster of similar populations, with somewhat smoother transitions.

CHAPTER 4

LEARNING IN SCRIP SYSTEMS

4.1 Introduction to Learning in Distributed Systems

Designers of distributed systems are frequently unable to determine how an agent in the system should behave, because optimal behavior depends on the user's preferences and the actions of others. A natural approach is to have agents use a learning algorithm. Many multiagent learning algorithms have been proposed including simple strategy update procedures such as *fictitious play* [28], multiagent versions of *Q-learning* [78], and *no-regret algorithms* [16].

However, existing algorithms are generally unsuitable for large distributed systems. In a distributed system, each agent has a limited view of the actions of other agents. Algorithms that require knowing, for example, the strategy chosen by every agent cannot be implemented. Furthermore, the size of distributed systems requires fast convergence. Users may use the system for short periods of time and conditions in the system change over time, so a practical algorithm for a system with thousands or millions of users needs to have a convergence rate that is sublinear in the number of agents. Existing algorithms tend to provide performance guarantees that are polynomial or even exponential. Finally, the large number of agents in the system guarantees that there will be noise. Agents will make mistakes and will behave in unexpectedly. Even if no agent changes his strategy, there can still be noise in agent payoffs. For example, a gossip protocol will match different agents from round to round; congestion in the underlying network may effect message delays between agents. A learning algorithm needs to be robust to this noise.

While finding an algorithm that satisfies these requirements for arbitrary games may be difficult, distributed systems have characteristics that make the problem easier. First, they involve a large number of agents. Having more agents may seem to make learning harder—after all, there are more possible interactions. However, it has the advantage that the outcome of an action typically depends only weakly on what other agents do. This makes outcomes robust to noise. Having a large number of agents also make it less useful for an agent to try to influence others; it becomes a better policy to try to learn an optimal response. In contrast, with a small number of agents, an agent can attempt to guide learning agents into an outcome that is beneficial for him.

Second, distributed systems are often *anonymous*; it does not matter *who* does something, but rather *how many* agents do it. For example, when there is congestion on a link, the experience of a single agent does not depend on who is sending the packets, but on how many are being sent. Anonymous games have a long history in the economics literature (e.g., [9]) and have been a subject of recent interest in the computer science literature [21, 31].

Finally, and perhaps most importantly, in a distributed system the system designer controls the game agents are playing. This gives us a somewhat different perspective than most work, which takes the game as given. We do not need to solve the hard problem of finding an efficient algorithm for all games. Instead, we can find algorithms that work efficiently for interesting classes of games, where for us “interesting” means “the type of games a system designer might wish agents to play.” Such games should be “well behaved,” since it would be strange to design a system where an agent’s decisions can influence other agents in pathological ways.

In Section 4.2, we show that *stage learning* [27] is robust, implementable with minimal information, and converges efficiently for an interesting class of games. In this algorithm, agents divide the rounds of the game into a series of stages. In each stage, the agent uses a fixed strategy except that he occasionally explores. At the end of a stage, the agent chooses as his strategy for the next stage whatever strategy had the highest average reward in the current stage. We prove that, under appropriate conditions, a large system of stage learners will follow (approximate) best-reply dynamics despite errors and exploration.

For games where best-reply dynamics converge, our theorem guarantees that learners will play an approximate Nash equilibrium. In contrast to previous results where the convergence guarantee scales poorly with the number of agents, our theorem guarantees convergence in a finite amount of time with an infinite number of agents. While the assumption that best-reply dynamics converge is a strong one, many interesting games converge under best-reply dynamics, including dominance solvable games and games with monotone best replies. Marden et al. [53] have observed that convergence of best-reply dynamics is often a property of games that humans design. Moreover, convergence of best-reply dynamics is a weaker assumption than a common assumption made in the mechanism design literature, that the games of interest have dominant strategies (each agent has a strategy that is optimal no matter what other agents do).

Simulation results, presented in Section 4.3, show that convergence is fast in practice: a system with thousands of agents can converge in a few thousand rounds. Furthermore, we identify two factors that determine the rate and quality of convergence. One is the number of agents: having more agents makes

the noise in the system more consistent so agents can learn using fewer observations. The other is giving agents statistical information about the behavior of other agents; this can speed convergence by an order of magnitude. Indeed, even noisy statistical information about agent behavior, which should be relatively easy to obtain and disseminate, can significantly improve performance.

While our theoretical results are limited to stage learning, they provide intuition about why other “well behaved” learning algorithms should also converge. Our simulations, which include two other learning algorithms, bear this out.

In Section 4.3.3 we demonstrate the effectiveness of stage learning in scrip systems. Our model of a scrip system from Section 2.1 is large and anonymous, but it is not a game in the sense used in this chapter. The major issue is that of state: in a scrip system, play in the current round depends on previous rounds through the amounts of money agents have. However, best-reply dynamics do converge, a fact that is central to the proof of Theorem 2.3.2.

Despite this limitation of our theoretical results, we show that stage learning can be adapted to this setting and that it allows agents to converge to equilibrium. Our results also demonstrate that stage learning is robust to factors such as churn (agents joining and leaving the system) and asynchrony (agents using stages of different lengths).

4.2 Theoretical Results

4.2.1 Large Anonymous Games

We are interested in anonymous games with countably many agents. Assuming that there are countably many agents simplifies the proofs; it is straightforward to extend our results to games with a large finite number of agents. Our model is adapted from that of [9]. Formally, a *large anonymous game* is characterized by a tuple $\Gamma = (\mathbb{N}, A, P, \text{Pr})$.

- \mathbb{N} is the countably infinite set of agents.
- A is a finite set of actions from which each agent can choose (for simplicity, we assume that each agent can choose from the same set of actions).
- $\Delta(A)$, the set of probability distributions over A , has two useful interpretations. The first is as the set of mixed actions. For $a \in A$ we will abuse notation and denote the mixed action that is a with probability 1 as a . In each round each agent chooses one of these mixed actions. The second interpretation of $\rho \in \Delta(A)$ is as the fraction of agents choosing each action $a \in A$. This is important for our notion of anonymity, which says an agent's utility should depend only on how many agents choose each action rather than who chooses it.
- $G = \{g : \mathbb{N} \rightarrow \Delta(A)\}$ is the set of (mixed) action profiles (i.e. which action each agent chooses). Given the mixed action of every agent, we want to know the fraction of agents that end up choosing action a . For $g \in G$, let $g(i)(a)$ denote the probability with which agent i plays a according to $g(i) \in \Delta(A)$. We can then express the fraction of agents in g that choose

action a as $\lim_{n \rightarrow \infty} (1/n) \sum_{i=0}^n g(i)(a)$, if this limit exists. If the limit exists for all actions $a \in A$, let $\rho_g \in \Delta(A)$ give the value of the limit for each a . The profiles g that we use are all determined by a simple random process. For such profiles g , the strong law of large numbers (SLLN) guarantees that with probability 1 ρ_g is well defined. Thus it will typically be well defined (using similar limits) for us to talk about the fraction of agents who do something.

- $P \subset \mathbb{R}$ is a finite set of payoffs agents can receive.
- $\text{Pr} : A \times \Delta(A) \rightarrow \Delta(P)$ denotes the distribution over payoffs that results when the agent performs action a and other agents follow action profile ρ . We use a probability distribution over payoffs rather than a payoff to model the fact that agent payoffs may change even if no agent changes his strategy. The expected utility of an agent who performs mixed action s when other agents follow action distribution ρ is $u(s, \rho) = \sum_{a \in A} \sum_{p \in P} p s(a) \text{Pr}_{a, \rho}(p)$. Our definition of Pr in terms of $\Delta(A)$ rather than G ensures the game is anonymous. We further require that Pr (and thus u) be *Lipschitz continuous*.¹ For definiteness, we use the L1 norm as our notion of distance when specifying continuity (the L1 distance between two vectors is the sum of the absolute values of the differences in each component). Note that this formulation assumes all agents share a common utility function.

An example of a large anonymous game is one where, in each round, each agent plays a two-player game against an opponent chosen at random. Such random matching games are common in the literature (e.g., [41]), and the mean-

¹Lipschitz continuity imposes the additional constraint that there is some constant K such that $|\text{Pr}(a, \rho) - \text{Pr}(a, \rho')| / \|\rho - \rho'\|_1 \leq K$ for all ρ and ρ' . Intuitively, this ensures that the distribution of outcomes doesn't change "too fast." This is a standard assumption that is easily seen to hold in the games that have typically been considered in the literature.

ing of "an opponent chosen at random" can be made formal [14]. In such a game, A is the set of actions of the two-player game and P is the set of payoffs of the game. Once every agent chooses an action, the distribution over actions is characterized by some $\rho \in \Delta(A)$. Let $p_{a,a'}$ denote the payoff for the agent if he plays a and the other agent plays a' . Then the utility of mixed action s given distribution ρ is

$$u(s, \rho) = \sum_{a, a' \in A^2} s(a) \rho(a') p_{a, a'}.$$

4.2.2 Best-Reply Dynamics

Given a game Γ and an action distribution ρ , a natural goal for an agent is to play the action that maximizes his expected utility with respect to ρ : $\operatorname{argmax}_{a \in A} u(a, \rho)$. We call such an action a *best reply* to ρ . In a practical amount of time, an agent may have difficulty determining which of two actions with close expected utilities is better, so we will allow agents to choose actions that are close to best replies. If a is a best reply to ρ , then a' is an η -*best reply* to ρ if $u(a', \rho) + \eta \geq u(a, \rho)$. There may be more than one η -best reply; we denote the set of η -best replies $ABR_\eta(\rho)$.

We do not have a single agent looking for a best reply; every agent is trying to find a one at the same time. If agents start off with some action distribution ρ_0 , after they all find a best reply there will be a new action distribution ρ_1 . We assume that $\rho_0(a) = 1/|A|$ (agents choose their initial strategy uniformly at random), but our results apply to any distribution used to determine the initial strategy. We say that a sequence (ρ_0, ρ_1, \dots) is an η -*best-reply sequence* if the support of ρ_{i+1} is a subset of $ABR_\eta(\rho_i)$; that is ρ_{i+1} gives positive probability only to

approximate best replies to ρ_i . A η best-reply sequence *converges* if there exists some t such that for all $t' > t$, $\rho_{t'} = \rho_t$. Note that this is a particularly strong notion of convergence because we require the ρ_t to converge in finite time and not merely in the limit. A game may have infinitely many best-reply sequences, so we say that *approximate best-reply dynamics converge* if there exists some $\eta > 0$ such that every η -best-reply sequence converges. The limit distribution ρ_t determines a mixed strategy that is an η -Nash equilibrium.

Our theorem shows that learners can successfully learn in large anonymous games where approximate best-reply dynamics converge. The number of stages needed to converge is determined by the number of best replies needed before the sequence converges. It is possible to design games that have long best-reply sequences, but in practice most games have short sequences. One condition that guarantees this is if ρ_0 and all the degenerate action distributions $a \in A$ (i.e., distributions that assign probability 1 to some $a \in A$) have unique best replies. In this case, there can be at most $|A|$ best replies before equilibrium is reached. Furthermore, in such games the distinction between η -best replies and best replies is irrelevant; for sufficiently small η , a η -best reply is a best reply. It is not hard to show that the property that degenerate strategies have unique best replies is generic; it holds for almost every game.

4.2.3 Stage Learners

An agent who wants to find a best reply may not know the set of payoffs P , the mapping from actions to distributions over payoffs \Pr , or the action distribution ρ (and, indeed, ρ may be changing over time), so he will have to use some type

of learning algorithm to learn it. Our approach is to divide the play of the game into a sequence of stages. In each stage, the agent almost always plays some fixed action a , but also explores other actions. At the end of the stage, he chooses a new a' for the next stage based on what he has learned. An important feature of this approach is that agents maintain their actions for the entire stage, so each stage provides a stable environment in which agents can learn. To simplify our results, we specify a way of exploring and learning within a stage (originally described in [27]), but our results should generalize to any “reasonable” learning algorithm used to learn within a stage. (We discuss what is “reasonable” in Chapter 5.) In this section, we show that, given a suitable parameter, at the each stage most agents will have learned a best reply to the environment of that stage.

Given a game Γ , in each round t agent i needs to select a mixed action $s_{i,t}$. Our agents use strategies that we denote a_ϵ , for $a \in A$, where $a_\epsilon(a) = 1 - \epsilon$ and $a_\epsilon(a' \neq a) = \epsilon/(|A| - 1)$. Thus, with a_ϵ , an agent almost always plays a , but with probability ϵ explores other strategies uniformly at random. Thus far we have not specified what information an agent can use to choose $s_{i,t}$. Different games may provide different information. All that we require is that an agent know all of his previous actions and his previous payoffs. More precisely, for all $t' < t$, he knows his action $a_{t'}(i)$ (which is determined by $s_{i,t'}$) and his payoffs $p_{t'}(i)$ (which is determined by $\Pr(a_{i,t'}, \rho_{t'})$, where $\rho_{t'}$ is the action distribution for round t' ; note that we do not assume that the agent knows $\rho_{t'}$.) Using this information, we can express the average value of an action over the previous $\tau = \lceil 1/\epsilon^2 \rceil$ rounds (the length of a stage).² Let $H(a, i, t) = \{t - \tau \leq t' < t \mid a_{t'}(i) = a\}$ be the set of recent rounds in which a was played by i . Then the average value

²The use of the exponent 2 is arbitrary. We require only that the expected number of times a strategy is explored increases as ϵ decreases.

is $V(a, i, t) = \sum_{i' \in H(a, i, t)} p_{i'}(i) / |H(a, i, t)|$ if $|H(a, i, t)| > 0$ and 0 otherwise. While we need the value of H only at times that are multiples of τ , for convenience we define it for arbitrary times t .

We say that an agent is an ϵ -stage learner if he chooses his actions as follows. If $t = 0$, s_t is chosen at random from $\{a_\epsilon \mid a \in A\}$. If t is a nonzero multiple of τ , $s_{i,t} = a(i, t)_\epsilon$ where $a(i, t) = \operatorname{argmax}_{a \in A} V(a, i, t)$. Otherwise, $s_{i,t} = s_{i,t-1}$. Thus, within a stage, his mixed action is fixed and at the end of a stage he updates it to use the action with the highest average value during the previous stage.

The evolution of a game played by stage learners is not deterministic; each agent chooses a random $s_{i,0}$ and the sequence of $a_t(i)$ and $p_t(i)$ he observes is also random. However, with a countably infinite set of agents, we can use the SLLN to make statements about the overall behavior of the game. Let $g_t(i) = s_{i,t}$. A run of the game consists of a sequence of triples (g_t, a_t, p_t) . The SLLN guarantees that with probability 1 the fraction of agents who choose a strategy a in a_t is $\rho_{g_t}(a)$. Similarly, the fraction of agents who chose a in a_t that receive payoff p will be $\Pr(a, \rho_{g_t})(p)$ with probability 1.

To make our notion of a stage precise, we refer to the sequence of tuples $(g_{n\tau}, a_{n\tau}, p_{n\tau}) \dots (g_{(n+1)\tau-1}, a_{(n+1)\tau-1}, p_{(n+1)\tau-1})$ as stage n of the run. During stage n there is a stationary action distribution that we denote $\rho_{g_{n\tau}}$. If $s_{i,(n+1)\tau} = a_\epsilon$ and $a \in ABR_\eta(g_{n\tau})$, then we say that agent i has *learned an η -best reply* during stage n of the run. As the following lemma shows, for sufficiently small ϵ , most agents will learn an η -best reply.

Lemma 4.2.1. *For all large anonymous games Γ , action profiles, approximations $\eta > 0$, and probabilities of error $e > 0$, there exists an $\epsilon^* > 0$ such that for $\epsilon < \epsilon^*$ and all n , if all agents are ϵ -stage learners, then at least a $1 - e$ fraction of agents will learn an η -best*

reply during stage n .

Proof. (Sketch) On average, an agent using strategy a_ϵ plays action a $(1 - \epsilon)\tau$ times during a stage and plays all other actions $\epsilon\tau/(n - 1)$ times each. For τ large, the realized number of times played will be close to the expectation value with high probability. Thus, if $\epsilon\tau$ is sufficiently large, then the average payoff from each action will be exponentially close to the true expected value (via a standard Hoeffding bound on sums of i.i.d. random variables), and thus each the learner will correctly identify an action with approximately the highest expected payoff with probability at least $1 - e$. By the SLLN, at least a $1 - e$ fraction of agents will learn an η -best reply. A detailed version of this proof in a more general setting can be found in [27]. \square

4.2.4 Convergence Theorem

Thus far we have defined large anonymous games where approximate best-reply dynamics converge. If all agents in the game are ϵ -stage learners, then the sequence $\hat{\rho}_0, \hat{\rho}_1, \dots$ of action distributions in a run of the game is not a best-reply sequence, but it is close. The action used by most agents most of the time in each $\hat{\rho}_n$ is the action used in ρ_n for some approximate best reply sequence.

In order to prove this, we need to define “close.” Our definition is based on the error rate e and exploration rate ϵ that introduces noise into $\hat{\rho}_n$. Intuitively, distribution $\hat{\rho}$ is close to ρ if, by changing the strategies of an e fraction of agents and having all agents explore an ϵ fraction of the time, we can go from an action profile with corresponding action distribution ρ to one with corresponding distribution $\hat{\rho}$. Note that this definition will not be symmetric.

In this definition, g identifies what (pure) action each agent is using that leads to ρ , g' allows an e fraction of agents to use some other action, and \hat{g} incorporates the fact that each agent is exploring, so each strategy is an a_ϵ (the agent usually plays a but explores with probability ϵ).

Definition 4.2.1. Action distribution $\hat{\rho}$ (e, ϵ) -close to ρ if there exist g, g' , and $\hat{g} \in G$ such that:

- $\rho = \rho_g$ and $\hat{\rho} = \rho_{\hat{g}}$;
- $g(i) \in A$ for all $i \in \mathbb{N}$;
- $\|\rho_g - \rho_{g'}\|_1 \leq 2e$ (this allows an e fraction of agents in g' to play a different strategy from g);
- for some $\epsilon' \leq \epsilon$, if $g'(i) = a$ then $\hat{g}(i) = a_{\epsilon'}$. ■

The use of ϵ' in the final requirement ensures that if two distributions are (e, ϵ) -close then they are also (e', ϵ') -close for all $e' \geq e$ and $\epsilon' \geq \epsilon$. As an example of the asymmetry of this definition, a_ϵ is $(0, \epsilon)$ close to a , but the reverse is not true. While (e, ϵ) -closeness is a useful distance measure for our analysis, it is an unnatural notion of distance for specifying the continuity of u , where we used the L1 norm. The following simple lemma shows that this distinction is unimportant; if $\hat{\rho}$ is sufficiently (e, ϵ) -close to ρ then it is close according to the L1 measure as well.

Lemma 4.2.2. If $\hat{\rho}$ is (e, ϵ) -close to ρ , then $\|\hat{\rho} - \rho\|_1 \leq 2(e + \epsilon)$.

Proof. Since $\hat{\rho}$ is (e, ϵ) -close to ρ , there exist g, g' , and \hat{g} as in Definition 4.2.1. Consider the distributions $\rho_g = \rho$, $\rho_{g'}$, and $\rho_{\hat{g}} = \hat{\rho}$. We can view these three distributions as vectors, and calculate their L1 distances. By Definition 4.2.1,

$\|\rho_g - \rho_{g'}\|_1 \leq 2e$. $\|\rho_{g'} - \rho_{\hat{g}}\|_1 \leq 2\epsilon$ because an ϵ fraction of agents explore. Thus by the triangle inequality, the L1 distance between ρ and $\hat{\rho}$ is at most $2(e + \epsilon)$. \square

We have assumed that approximate best reply sequences of ρ_n converge, but during a run of the game agents will actually be learning approximate best replies to $\hat{\rho}_n$. The following lemma shows that this distinction does not matter if ρ and $\hat{\rho}$ are sufficiently close.

Lemma 4.2.3. *For all η there exists a d_η such that if $\hat{\rho}$ is (e, ϵ) -close to ρ , $e > 0$, $\epsilon > 0$, and $e + \epsilon < d_\eta$ then $ABR_{(\eta/2)}(\hat{\rho}) \subseteq ABR_\eta(\rho)$.*

Proof. Let K be the maximum of the Lipschitz constants for all $u(a, \cdot)$ and $d_\eta = \eta/(8K)$. Then for all $\hat{\rho}$ that are (e, ϵ) -close to ρ and all a , $|u(a, \hat{\rho}) - u(a, \rho)| \leq \|\hat{\rho} - \rho\|_1 K \leq 2\eta/(8K)K = \eta/4$ by Lemma 4.2.2.

Let $a \notin ABR_\eta(\rho)$ and $a' \in \operatorname{argmax}_{a' \in ABR_\eta(\rho)} u(a', \hat{\rho})$. Then $u(a, \rho) + \eta < u(a', \rho)$. Combining this with the above gives $u(a, \hat{\rho}) + \eta/2 < u(a', \hat{\rho})$. Thus $a \notin ABR_{\eta/2}(\hat{\rho})$. \square

Lemmas 4.2.1 and 4.2.3 give requirements on (e, ϵ) . In the statement of the theorem, we call (e, ϵ) η -acceptable if they satisfy the requirements of both lemmas for $\eta/2$ and all η -best-reply sequences converge in Γ .

Theorem 4.2.1. *Let Γ be a large anonymous game where approximate best-reply dynamics converge and let (e, ϵ) be η -acceptable for Γ . If all agents are ϵ -stage learners then, for all runs, there exists an η -best-reply sequence ρ_0, ρ_1, \dots such that in stage n at least a $1 - e$ fraction will learn a best reply to ρ_n with probability 1.*

Proof. $\rho_0 = \hat{\rho}_0$, so $\hat{\rho}_0$ is (e, ϵ) -close to ρ . Assume $\hat{\rho}_n$ is (e, ϵ) -close to ρ . By

Lemma 4.2.1 at least a $1 - e$ fraction will learn a $\eta/2$ -best reply to $\hat{\rho}_n$. By Lemma 4.2.3, this is a η -best reply to ρ_n . Thus $\hat{\rho}_{n+1}$ will be (e, ϵ) -close to ρ_{n+1} . \square

Theorem 4.2.1 guarantees that after a finite number of stages, agents will be close to an approximate Nash equilibrium profile. Specifically, $\hat{\rho}_n$ will be (e, ϵ) -close to an η -Nash equilibrium profile ρ_n . Note that this means that $\hat{\rho}_n$ is actually an η' -Nash equilibrium for a larger η' that depends on η, e, ϵ , and the Lipschitz constant K .

Our three requirements for a practical learning algorithm were that it require minimal information, converge quickly in a large system, and be robust to noise. Stage learning requires only that an agent know his own payoffs, so the first condition is satisfied. Theorem 4.2.1 shows that it satisfies the other two requirements. Convergence is guaranteed in a finite number of stages. While the number of stages depends on the game, in Section 4.2.2 we argued that in many cases it will be quite small. Finally, robustness comes from tolerating an e fraction of errors. While in our proofs we assumed these errors were due to learning, the analysis is the same if some of this noise is from other sources such as churn or agents making errors. We discuss this issue more in Chapter 5.

4.3 Simulation Results

In this section, we discuss experimental results that demonstrate the practicality of learning in large anonymous games. Theorem 4.2.1 guarantees convergence for a sufficiently small exploration probability ϵ , but decreasing ϵ also increases τ , the length of a stage. Our first set of experiments shows that the necessary

values of ϵ and τ are quite reasonable in practice. While our theorem applies only to stage learning, the analysis provides intuition as to why a reasonable algorithm that changes slowly enough that other learners have a chance to learn best replies should converge as well. To demonstrate this, we also implemented two other learning algorithms, which also quickly converged.

Our theoretical results make two significant predictions about factors that influence the rate of convergence. Lemma 4.2.1 tells us that the length of a stage is determined by the number of times each strategy needs to be explored to get an accurate estimate of its value. Thus, the amount of information provided by each observation has a large effect on the rate of convergence. For example, in a random matching game, an agent's payoff provides information about the strategy of one other agent. On the other hand, if he receives his expected payoff for being matched, a single observation provides information about the entire distribution of strategies. In the latter case the agent can learn with many fewer observations. A related prediction is that having more agents will lead to faster convergence, particularly in games where payoffs are determined by the average behavior of other agents, because variance in payoffs due to exploration and mistakes decreases as the number of agents increases. Our experimental results illustrate both of these phenomena.

The game used in our first set of experiments, like many simple games used to test learning algorithms, is symmetric. Hopkins [41] showed that many learning algorithms are well behaved in symmetric games with large populations. To demonstrate that our main results are due to something other than symmetry, we also tested our stage learning on an asymmetric game, and observed convergence even with a small population.

Finally, after exploring stage learning in simple games, we implemented a variant of stage learning for scrip systems. To demonstrate the applicability of this approach to real systems, we included experiments where there is churn (agents leaving and being replaced by new agents) and agents learning at different rates.

4.3.1 A Contribution Game

In our first set of experiments, agents play a contribution game (also called a Diamond-type search model in [57]). In the contribution game, two agents choose strategies from 0 to 19, indicating how much effort they contribute to a collective enterprise. The value to an agent depends on how much he contributes, as well as how much the other agent contributes. If he contributes x and the contribution of the other agent is y , then his utility is $4xy - (x - 5)^3$. In each round of our game, each agent is paired with a random agent and they play the contribution game.

We implemented three learning algorithms to run on this game. Our implementation of stage learners is as described in Section 4.2.3, with $\epsilon = 0.05$. Rather than taking the length of stage τ as $1/\epsilon^2$, we set $\tau = 2500$ to improve performance. Our second algorithm is based on that of Hart and Mas-Colell [37], with improvements suggested by Greenwald et al. [32]. This algorithm takes parameters M and δ (the exploration probability). We used $M = 16$ and $\delta = 0.05$. Our final learning algorithm is Exp3 [5]. We set γ , the exploration probability, to 0.05. This algorithm requires that payoffs be normalized to lie in $[0, 1]$. Since a few choices of strategies lead to very large negative payoffs, a naive normaliza-

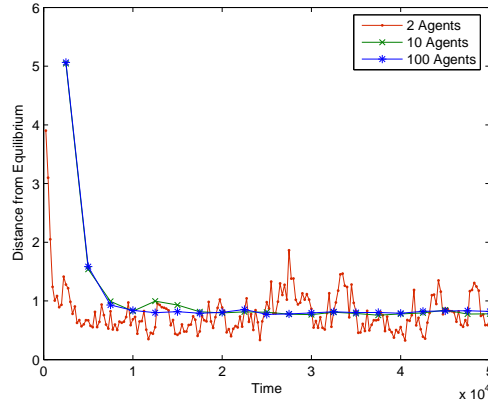


Figure 4.1: Stage Learners with Random Matching

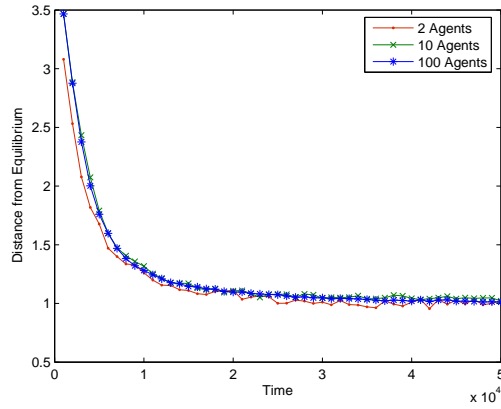


Figure 4.2: Hart and Mas-Colell with Random Matching

tion leads to almost every payoff being close to 1. For better performance, we normalized payoffs such that most payoffs fell into the range $[0, 1]$ and any that were outside were set to 0 or 1 as appropriate.

The results of these three algorithms are shown in Figures 4.1, 4.2, and 4.3. Each curve shows the distance from equilibrium as a function of the number of rounds of a population of agents of a given size using a given learning algorithm. The results were averaged over 10 runs. Since the payoffs for nearby strategies are close, we want our notion of distance to take into account that

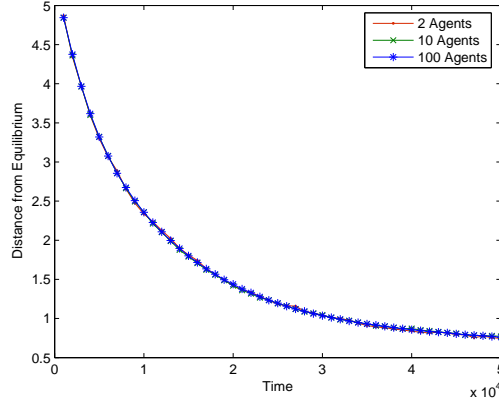


Figure 4.3: Exp3 with Random matching

agents playing 7 are closer to equilibrium (8) than those playing zero. Therefore, we consider the expected distance of ρ from equilibrium: $\sum_a \rho(a)|a - 8|$. To determine ρ , we counted the number of times each action was over the length of a stage, so in practice the distance will never be zero due to mistakes and exploration. For ease of presentation, the graph shows only populations of size up to 100; similar results were obtained for populations up to 5000 agents.

For stage learning, increasing the population size has a dramatic impact. With two agents, mistakes and best replies to the results of these mistakes cause behavior to be quite chaotic. With ten agents, agents successfully learn, although mistakes and suboptimal strategies are quite frequent. With one hundred agents, all the agents converge quickly to near equilibrium strategies and significant mistakes are rare.

Despite a lack of theoretical guarantees, our other two algorithms also converge, although somewhat more slowly. The performance of Exp3 is essentially identical to stage learning. Hart and Mas-Colell's algorithm only has asymptotic convergence guarantees, and even with optimized tends to converge very

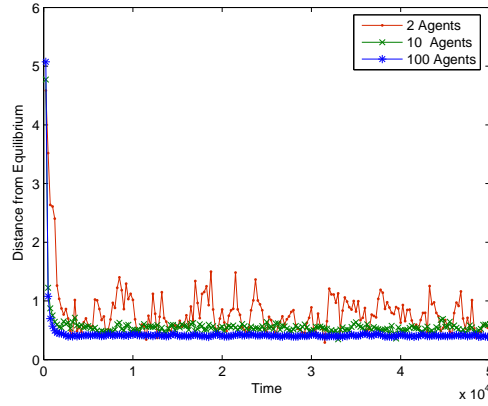


Figure 4.4: Stage Learning with Average-Based Payoffs

slowly in practice if tuned for tight convergence. So to get it to converge in a reasonable amount of time we tuned the parameters to accept somewhat weaker convergence (although for the particular game shown here the difference in convergence is not dramatic).

Convergence of stage learning in the random-matching game takes approximately 10,000 rounds, which is too slow for many applications. If a system design requires this type of matching, this makes learning problematic. However, the results of Figure 4.4 suggest that the learning could be done much faster if the system designer could supply agents with more information. This suggests that collecting statistical information about the behavior of agents may be a critical feature for ensuring fast convergence. To model such a scenario, consider a related game where, rather than being matched against a random opponent, all agents contribute to the same project and their reward is based on the average contribution of the other agents. The results of stage learning in this game are shown in Figure 4.4. With so much more information available to agents from each observation, we were able to cut the length of a stage by a factor of 10. The number of stages needed to reach equilibrium remained essentially the

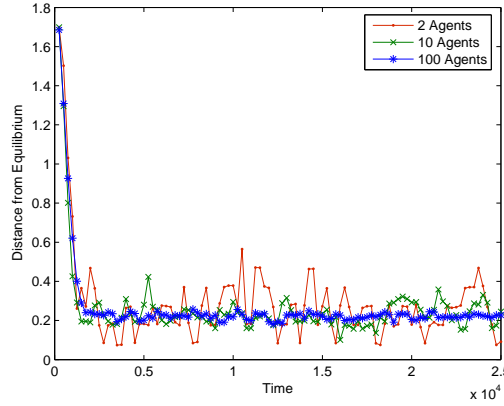


Figure 4.5: Stage Learners in a Congestion Game

same. Convergence was tighter as well; mistakes were rare and almost all of the distance from equilibrium is due to exploration.

4.3.2 A Congestion Game

For a different game, we tested the performance of stage learners in a congestion game. This game models a situation where two agents share a network link. They gain utility proportional to their transmission rate over the link, but are penalized based on the resulting congestion they experience. The game is asymmetric because the two different types of agents place different values on transmission rate. The game is described in detail in [32].

Figure 4.5 shows that stage learners were able to learn very quickly in this game, using stages of length 250 even though they were being randomly matched against a player of the other type. Because the different types of agents had different equilibrium strategies, the distance measure we use is to treat the observed distribution of strategies and the equilibrium distribution as vectors

and compute their L1 distance.

4.3.3 Scrip Systems

To demonstrate how stage learning could be applied in a scrip system, we tested a variant of stage learners in the model of a scrip system from Section 2.1. Recall that, in the model, agents pay other agents to provide them service and in turn provide service themselves to earn money to pay for future service. Agents may place different values on receiving service (γ), incur different costs to provide service (α), discount future utility at different rates (δ), and have different availabilities to provide service (β). We used a single type of agent with parameters $\gamma = 1.0$, $\alpha = 0.05$, $\delta = 0.9$, $\beta = 1$, average amount of money per agent $m = 1$, and stages of 200 rounds per agent (only one agent makes a request each round).

This model is not a large anonymous game because whether an agent should provide service depends on how much money he currently has. Thus, stage learning as specified doesn't work, because it does not take into account the current state of the (stochastic) game. Despite this, we can still implement a variant of stage learning: fix a strategy during each stage and then at the end of the stage use an algorithm designed for this game to determine a new strategy that is a best reply to what the agent observed. Our algorithm works by estimating the agent's probabilities of making a request and being chosen as a volunteer in each round, and then uses these probabilities to compute an optimal policy. Figure 4.6 shows that this is quite effective. The distance measure used is based on directly measuring the distance of the agent's chosen (thresh-

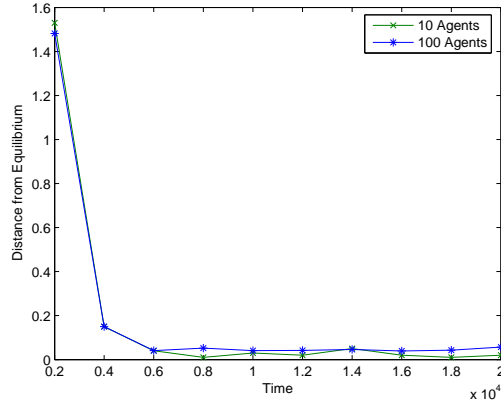


Figure 4.6: Stage Learners in a Scrip System

old) strategy from the equilibrium strategy, since unlike the previous games it is impossible to directly infer the agent's strategy in each round solely from his decision whether or not to volunteer. Note that the number of rounds has been normalized based on the number of agents in Figure 4.6 and later figures; stages actually lasted ten times as long with 100 agents.

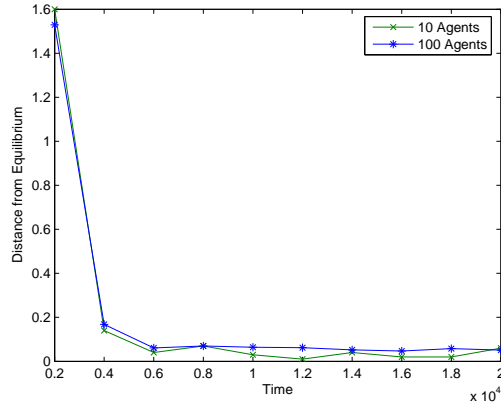


Figure 4.7: A Scrip System with Churn

Real systems do not have a static population of learning agents. To demonstrate the robustness of stage learning to churn, we replaced ten percent of the agents with new agents with randomly chosen initial strategies at the end of

each period. As Figure 4.7 shows, this has essentially no effect on convergence.

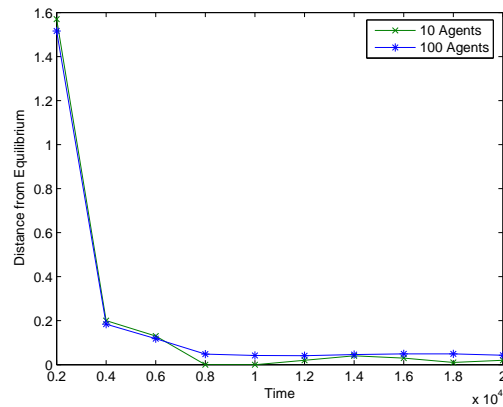


Figure 4.8: A Scrip System with Different Stage Lengths

Finally, in a real system it is often unreasonable to expect all agents to be able to update their strategies at the same time. Figure 4.8 shows that having half the agents use stages of 222 rounds per agent rather than 200 did not have a significant effect on convergence.

CHAPTER 5

CONCLUSION

We have given a formal analysis of a scrip system and have shown that approximate equilibria exist in threshold strategies and that the distribution of money in these equilibria is given by relative entropy. As part of our equilibrium argument, we have shown that the best-reply function is monotone. This proves the existence of equilibria in pure strategies and permits efficient algorithms to compute these equilibria. We have also examined some of the practical implications of these theoretical results. For those interested in studying the agents of scrip systems, our characterization of equilibrium distribution of money forms the basis for techniques relevant to inferring characteristics of the agents of a scrip system from the distribution of money. For a system designer, our results on optimizing the money supply provide a simple maxim: keep adding money until the system is about to experience a monetary crash.

We have also seen that our model can be used to understand the effects of nonstandard agent behavior on a scrip system. It provides insight into the effects of altruists and hoarders on a scrip system and guidance to system designers for dealing with them (less and more money respectively). Sybils are generally bad, but can typically be discouraged by imposing a moderate cost and possibly biasing the process for selecting a volunteer. On the other hand, collusion tends to be a net benefit and should be encouraged. Indeed, the entire purpose of the system is to allow users to collude and provide each other with service despite incentives to free ride.

Our model makes a number of assumptions that are worthy of further discussion. Some of the simplifying assumptions can be relaxed without signifi-

cant changes to our results (albeit at the cost of greater strategic and analytic complexity). At a high level, our results show the system converges to a steady state when agents follow simple threshold strategies and that there is in fact an equilibrium in these strategies. If, for example, rather than all requests having the same value to agent (γ_i), the value of a request is stochastic, agents might wish to have thresholds for each type of request. This would allow an agent to forgo a low-valued request if he is low on money. This makes the space of agent strategies larger and significantly complicates the proofs in the appendix, but this high-level characterization still holds.

The most significant assumption we make is that prices are fixed. However, our results provide insight even if we relax this assumption. With variable prices, the behavior of the system depends on the value of β , the probability that an agent can satisfy a request. For large β , where there are a large number of agents who can satisfy each request, we expect the resulting competition to effectively produce a fixed price, so our analysis applies directly. For small β , where there are few volunteers for each request, variable prices can have a significant impact.

However, allowing prices to be set endogenously, by bidding, has a number of negative consequences. For one thing, it removes the ability of the system designer to optimize the system using monetary policy. In addition, for small β , it is possible for colluding agents to form a cartel to fix prices on a resource they control. It also greatly increases the strategic complexity of using the system: rather than choosing a single threshold, agents need an entire pricing scheme. Finally, the search costs and costs of executing a transaction are likely to be higher with variable prices. Thus, we believe that adopting a fixed price or a

small set of fixed prices is often a reasonable design decision.

We believe there is often a happy medium between a single, permanent fixed price and prices that change freely from round to round; indeed, our advice to system designers points naturally toward it. In particular, our advice about how to optimize the amount of money relies on experimentation and observation to determine what agents are doing and what their utilities are. This information then tells the designer how much money she should provide. Since adjusting the amount of money is equivalent to adjusting prices, the designer could incorporate this process into a price setting rule. Depending on the nature of the system, this could either be done manually over time (if the information is difficult to gather and analyze) or automatically (if the information gathering and analysis can itself be automated). From this perspective, a monetary crash, though real, is not something to be feared. Instead, it is just a strong signal that the current price, while probably not too far off from a very good price, requires adjustment. Naturally, this relies on a process that proceeds slow enough that agents myopically ignore the effects of future price changes in determining their current action.

Our model provides only the beginning of a full understanding of scrip systems. Many interesting open questions remain for future work. To name a few:

- Our model makes a number of strong predictions about the agent strategies, distribution of money, and effects of variations in the money supply. It also provides techniques to help analyze characteristics of agents of a scrip system. It would be interesting to test these predictions on a real functioning scrip system to either validate the model or gain insight from where its predictions are incorrect.

- In many systems there are overlapping communities of various sizes that are significantly more likely to be able to satisfy each other's requests. For example, in a P2P filesharing system, people are more likely to be able to satisfy the requests of others who share the same interests. It would be interesting to investigate the effect of such communities on the equilibrium of our system.
- It seems unlikely that altruism and hoarding are the only two types of "irrational" behavior we will find in real systems. Are there other major types that our model can provide insight into? Furthermore, it seems natural that the behavior of a very small group of agents should not be able to change the overall behavior of the system. Can we prove results about equilibria and utility when a small group follows an arbitrary strategy? This is particularly relevant when modeling attackers. See [1] for general results in this setting.

We have also explored the problem of how agents can find equilibria in a scrip system, as well as in distributed systems more generally. Learning in distributed systems requires algorithms that are scalable to thousands of agents and can be implemented with minimal information about the actions of other agents. Most general-purpose multiagent learning algorithms fail one or both of these requirements. We have shown here that stage learning can be an efficient solution in large anonymous games where approximate best-reply dynamics lead to approximate pure strategy Nash equilibria. Scrip systems seem to satisfy these assumptions well enough that stage learning works in practice.

Our results also highlight two factors that aid convergence. First, having more learners often improves performance. With more learners, the noise in-

troduced into payoffs by exploration and mistakes becomes more consistent. Second, having more information typically improves performance. Publicly available statistics about the observed behavior of agents can allow an agent to learn effectively while making fewer local observations. Our simulations demonstrate the effects of these two factors, as well how our results generalize to situations with other learning algorithms, churn, and asynchrony. However, there are many further issues that merit exploration.

Other Learning Algorithms

Our theorem assumes that agents use a simple rule for learning within each stage: they average the value of payoffs received. However, there are certainly other rules for estimating the value of an action; any of these can be used as long as the rule guarantees that errors can be made arbitrarily rare given sufficient time. It is also not necessary to restrict agents to stage learning. Stage learning guarantees a stationary environment for a period of time, but such strict behavior may not be needed or practical. Other approaches, such as exponentially discounting the weight of observations [32, 54] or Win or Learn Fast [13] allow an algorithm to focus its learning on recent observations and provide a stable environment in which other agents can learn.

Other Update Rules

In addition to using different algorithms to estimate the values of actions, a learner could also change the way he uses those values to update his behavior. For example, rather than basing his new strategy on only the last stage, he could

base it on the entire history of stages and use a rule in the spirit of fictitious play. Since there are games where fictitious play converges but best-reply dynamics do not, this could extend our results to another interesting class of games, as long as the errors in each period do not accumulate over time. Another possibility is to update probabilistically or use a tolerance to determine whether to update (see e.g. [25, 37]). This could allow convergence in games where best-reply dynamics oscillate or decrease the fraction of agents who make mistakes once the system reaches equilibrium.

Model Assumptions

Our model makes several unrealistic assumptions, most notably that there are countably many agents who all share the same utility function. Essentially the same results holds with a large, finite number of agents, adding a few more “error terms”. In particular, since there is always a small probability that every agent makes a mistake at the same time, we can prove only that no more than a $1 - \epsilon$ fraction of the agents make errors in most rounds, and that agents spending most of their time playing equilibrium strategies.

We have also implicitly assumed that the set of agents is fixed. As Figure 4.7 shows, we could easily allow for churn. A natural strategy for newly arriving agents is to pick a random a_ϵ to use in the next stage. If all agents do this, it follows that convergence is unaffected: we can treat the new agents as part of the ϵ fraction that made a mistake in the last stage. Furthermore, this tells us that newly arriving agents “catch up” very quickly. After a single stage, new agents are guaranteed to have learned a best reply with probability at least $1 - \epsilon$.

Finally, we have assumed that all agents have the same utility function. Our results can easily be extended to include a finite number of different types of agents, each with their own utility function, since the SLLN can be applied to each type of agent. We believe that our results hold even if the set of possible types is infinite. This can happen, for example, if an agent’s utility depends on a valuation drawn from some interval. However, some care is needed to define best-reply sequences in this case.

State

One common feature of distributed systems not addressed in our theoretical results on scrip systems is state. As in scrip systems, where an agent’s decision about whether to volunteer depends on how much money he currently has, an agent’s current state is often an important factor in choosing an optimal action.

In principle, we could extend our framework to games with state: in each stage each agent chooses a policy to usually follow and explores other actions with probability ϵ . Each agent could then use some *off-policy algorithm* (one where the agent can learn without controlling the sequence of observations; see [47] for examples) to learn an optimal policy to use in the next stage. One major problem with this approach is that standard algorithms learn too slowly for our purposes. For example, Q-learning [78] typically needs to observe each state-action pair hundreds of times in practice. The low exploration probability means that the expected $|S||A|/\epsilon$ rounds needed to explore each state action pair even once is large. Efficient learning requires more specialized algorithms that can make better use of the structure of a problem. However, the use of specialized algorithms makes providing a general guarantee of convergence more

difficult. Another problem is that, even if an agent explores each action for each of his possible local states, the payoff he receives will depend on the states of the other agents, and thus the actions they chose. We need some property of the game to guarantee that this distribution of states is in some sense “well behaved.” This is the case with scrip systems, where the states of agents converge to a distribution defined using relative entropy, but it is not clear how this generalizes.

Mixed Equilibria

Another restriction of our results is that our agents only learn pure strategies. One way to address this is to discretize the mixed strategy space (see e.g. [25]). If one of the resulting strategies is sufficiently close to an equilibrium strategy and best-reply dynamics converge with the discretized strategies, then we expect agents to converge to a near-equilibrium distribution of strategies. We have had empirical success using this approach to learn to play rock-paper-scissors.

Unexpected and Byzantine Behavior

In practice, we expect that not all agents will be trying to learn optimal behavior in a large system. Some agents may simply play some particular (possibly mixed) strategy that they are comfortable with, without trying to learn a better strategy. Others may be learning but with an unanticipated utility function. Whatever their reasons, if these sufficiently few such agents are choosing their strategies i.i.d. from fixed distributions (or at least fixed for each stage), then our results hold without change. This is because we already allow an ϵ frac-

tion of agents to make arbitrary mistakes, so we can treat these agents as simply mistaken. This means that altruists, hoarders, sybils, and collusion can be tolerated by the learning algorithm, since in all these cases agents use a consistent threshold strategy.

We might want to go further, and tolerate agents who do not even follow a consistent strategy for an entire stage. Tolerating such agents would make a learning strategy robust to Byzantine behavior. In games with a Lipschitz-continuous utility function, the actions of a small enough fraction of agents cannot cause a large change in the expected utilities of other agents. Thus, such games are typically robust to Byzantine behavior [31].

APPENDIX A

APPENDIX

A.1 Proof of Theorem 2.2.1

Given a Markov chain \mathcal{M} over a state space X and state $s \in \mathcal{S}$, let $I_{\vec{x}, \vec{y}}^r$ be the random variable that is 1 if \mathcal{M} is in state \vec{y} at time r and the chain started in state \vec{x} and 0 otherwise. Then $\lim_{r \rightarrow \infty} \Pr(I_{\vec{x}, \vec{y}}^r = 1)$ is the limit probability of being in state \vec{y} given that the Markov chain starts in state \vec{x} . In general, this limit does not exist. However, there are well-known conditions under which the limit exists, and is independent of the initial state \vec{x} . A Markov chain is said to be *irreducible* if every state is reachable from every other state; it is *aperiodic* if, for every state \vec{x} , there exist two cycles from \vec{x} to itself such that the gcd of their lengths is 1.

Theorem A.1.1. [64] *If \mathcal{M} is a finite, irreducible, and aperiodic Markov chain over state space X , then there exists a d such that, for all \vec{x} and $\vec{y} \in X$, $\lim_{r \rightarrow \infty} \Pr(I_{\vec{x}, \vec{y}}^r = 1) = d$.*

Thus, if we can show that \mathcal{M} is finite, irreducible, and aperiodic, then the limit distribution exists and is independent of the start state \vec{x} . This is shown in the following lemma.

Lemma A.1.1. *If there are at least three agents, then \mathcal{M} is finite, irreducible, and aperiodic and therefore has a limit distribution π .*

Proof. \mathcal{M} is clearly finite since X is finite. We prove that it is irreducible by showing that state \vec{y} is reachable from state \vec{x} by induction on the distance $w = \sum_{i=1}^n |x_i - y_i|$ (i.e., the sum of the absolute differences in the amount of money each person has in states \vec{x} and \vec{y}). If $w = 0$, then $\vec{x} = \vec{y}$ so we are done. Suppose that

$w > 0$ and all pairs of states that are less than w apart are reachable from each other. Consider a pair of states \vec{x} and \vec{y} such that the distance from \vec{x} to \vec{y} is w . Since $w > 0$ and the total amount of money is the same in all states, there must exist i_1 and i_2 such that $x_{i_1} > y_{i_1}$ and $x_{i_2} < y_{i_2}$. Thus, in state \vec{y} , i_1 is willing to work (since he has strictly less than the threshold amount of money) and i_2 has money to pay him (since i_2 has a strictly positive amount of money). The state \vec{z} that results from i_1 doing work for i_2 in state \vec{y} is of distance $w - 2$ from \vec{x} . By the induction hypothesis, \vec{z} is reachable from \vec{x} . Since \vec{y} is clearly reachable from \vec{z} , \vec{y} is reachable from \vec{x} .

Finally, we must show that \mathcal{M} is aperiodic. Suppose \vec{x} is a state such that there exist three agents i_1, i_2 , and i_3 where i_1 has more than 0 dollars and i_2 and i_3 have less than their threshold amount of money. There must be such a state by our assumption that m is “interesting.” Clearly there is a cycle of length 2 from \vec{x} to itself: i_2 does work for i_1 and then i_2 does work for i_1 . There is also a cycle of length 3: i_2 does work for i_1 , i_3 does work for i_2 , then i_1 does work for i_3 . □

We next give an explicit formula for the limit distribution. Recall that in the special case discussed in the main text, β_t, χ_t , and ρ_t were the same for all types, so the transition probabilities were symmetric and the limit distribution was uniform. While with more general values they are no longer symmetric, they still have significant structure that allows us to give a concise description of the limit distribution.

Lemma A.1.2. *For all states \vec{x} of \mathcal{M} , let $w_{\vec{x}} = \prod_i (\beta_{\tau(i)} \chi_{\tau(i)} / \rho_{\tau(i)})^{x_i}$, and let $Z = \sum_{\vec{y}} w_{\vec{y}}$. Then the limit distribution of \mathcal{M} is $\pi(\vec{x}) = w_{\vec{x}} / Z$.*

Proof. Define π by taking $\pi(\vec{x}) = w_{\vec{x}}/Z$, where $w_{\vec{x}}$ and Z are as in the statement of the lemma. If $T_{\vec{x}\vec{y}}$ is the probability of transitioning from state \vec{x} to state \vec{y} , it is well known that it suffices to show that π satisfies the *detailed balance condition* [64], i.e., $\pi(\vec{x})T_{\vec{x}\vec{y}} = \pi(\vec{y})T_{\vec{y}\vec{x}}$ for all states \vec{x} and \vec{y} and π is a probability measure. The fact that π is a probability measure is immediate from its definition. To check the first condition, let \vec{x} and \vec{y} be adjacent states such that \vec{y} is reached from \vec{x} by i spending a dollar and j earning a dollar. This means that for the transition from \vec{x} to \vec{y} to happen, i must be chosen to spend a dollar and j must be able to work and chosen to earn the dollar. Similarly for the reverse transition to happen, j must be chosen to spend a dollar and i must be able to work and chosen to earn the dollar. All other agents have the same amount of money in each state, and so will make the same decision in each state. Thus the probabilities associated with each transition differ only in the relative likelihoods of i and j being chosen at each point. These may differ for three reasons: one might be more likely to be able to satisfy requests (β), to want to make requests (ρ), or to be chosen to satisfy requests (χ). Thus, for some p , which captures the effect of other agents volunteering on the likelihood of i and j being chosen, we can write the transition probabilities as $T_{\vec{x}\vec{y}} = \rho_{\tau(i)}\beta_{\tau(j)}\chi_{\tau(j)}p$ and $T_{\vec{y}\vec{x}} = \rho_{\tau(j)}\beta_{\tau(i)}\chi_{\tau(i)}p$. From the definition of π , we have that

$$\frac{\pi(\vec{x})}{\pi(\vec{y})} = \frac{\beta_{\tau(i)}\chi_{\tau(i)}\rho_{\tau(j)}}{\rho_{\tau(i)}\beta_{\tau(j)}\chi_{\tau(j)}} = \frac{T_{\vec{y}\vec{x}}}{T_{\vec{x}\vec{y}}}.$$

Thus, $\pi(\vec{x})T_{\vec{x}\vec{y}} = \pi(\vec{y})T_{\vec{y}\vec{x}}$, as desired. \square

Note that for the special case considered in the main text, Lemma A.1.2 shows that the limit distribution is the uniform distribution.

The limit distribution tells us the long run probability of being in a given state. Theorem 2.2.1 does not mention states directly, but rather the distributions

of money associated with a state. In order to prove the theorem, we need to know the probability of being in some state associated with a given distribution. This is established in the following lemma.

Lemma A.1.3. *Let π be the limit distribution from Lemma A.1.2, and let $V(d) = H(d) - H(\vec{f}) - \log Z + \sum_t \sum_{i=0}^{k_t} id(t, i) \log \omega_t$ (where H is the standard entropy function; that is, $H(d) = \sum_{t,i} d(t, i) \log d(t, i)$). For all $d \in \Delta_{\vec{f}, m, \vec{k}}$, either $\pi(\{\vec{x} \mid d^{\vec{x}} = d\}) = 0$ or $F(hn)e^{hnV(d)} \leq \pi(\{\vec{x} \mid d^{\vec{x}} = d\}) \leq G(hn)e^{hnV(d)}$, where F and G are polynomials.*

Proof. Before computing the probability of being in such a state, we first compute the number of states. It is possible that there is no state \vec{x} such that $d = d^{\vec{x}}$ (e.g., if hn is odd and d has half the agents with 0 dollars). If there is such a state \vec{x} , each such state has $hnd(t, i)$ agents of type t with i dollars. Thus, the number of states \vec{x} with $d = d^{\vec{x}}$ is the number of ways to divide the agents into groups of these sizes. Since there are hnf_t agents of type t , the number of such states is

$$\prod_t \binom{hnf_t}{hnd(t, 0), \dots, hnd(t, k_t)}.$$

To complete the proof, we use the fact (shown in the proof of Lemma 3.11 of [33]) that

$$\frac{1}{F(hn)} e^{hnf_t H(d_t)} \leq \binom{hnf_t}{hnd(t, 0), \dots, hnd(t, k_t)} \leq G(hn) e^{hnf_t H(d_t)},$$

where F and G are polynomial in hn , and d_t is the distribution restricted to a single type t (i.e., $d_t(i) = d(t, i) / \sum_i d(t, i)$). The (generalized) grouping property [20] of entropy allows us to express $H(d)$ in terms of the entropy of the distributions for each fixed t , or the $H(d_t)$. Because $f_t = \sum_i d(t, i)$, this has the particularly simple form $H(d) = H(\vec{f}) + \sum_t f_t H(d_t)$. Thus, up to a polynomial factor, the number of such states is

$$\prod_t e^{hnf_t H(d_t)} e^{hn(\sum_t f_t H(d_t))} = e^{hn(H(d) - H(\vec{f}))}.$$

By Lemma A.1.2, each of these states has the same probability $\pi(\vec{x})$. Thus, dropping the superscript \vec{x} on $d^{\vec{x}}$ for brevity, the probability of being in such a state is:

$$\begin{aligned}
e^{hn(H(d)-H(\vec{f}))}\pi(\vec{x}) &= e^{hn(H(d)-H(\vec{f}))} \prod_i (\beta_i \chi_i / \rho_i)^{x_i} / Z \\
&= e^{hn(H(d)-H(\vec{f}))} Z^{-hn} \prod_i (\omega_{t_i})^{x_i} \\
&= e^{hn(H(d)-H(\vec{f}))} Z^{-hn} \prod_t \prod_{i=0}^{k_t} (\omega_t)^{h_{nid}(t,i)} \\
&= e^{hn(H(d)-H(\vec{f}))} Z^{-hn} \prod_t \prod_{i=0}^{k_t} e^{h_{nid}(t,i) \log \omega_t} \\
&= e^{hn(H(d)-H(\vec{f})) - \log Z + \sum_t \sum_{i=0}^{k_t} id(t,i) \log \omega_t} \\
&= e^{hnV(d)}
\end{aligned}$$

□

Theorem 2.2.1 says that there exists a $q \in \Delta_{\vec{f},m,\vec{k}}$ (i.e., a probability distribution on agent types t and amounts of money i) with certain properties. We now define the appropriate q . Let

$$q(t, i) = (\omega_t)^i / \left(\sum_t \sum_{j=0}^{k_t} (\omega_t)^j \right). \quad (\text{A.1})$$

It is not immediately clear why this is the right choice of q . As the following lemma shows, this definition allows us to characterize the distribution that maximizes the probability of being in a state corresponding to that distribution (as given by Lemma A.1.3) in terms of relative entropy.

Lemma A.1.4. *The unique maximum of $V(d) = H(d) - H(\vec{f}) - \log Z + \sum_t \sum_{i=0}^{k_t} id_i^t \log \omega_t$ on $\Delta_{\vec{f},m,\vec{k}}$ occurs at d_q^* .*

Proof. For brevity, we drop the superscript \vec{x} on d and let $Y = \sum_t \sum_i (\omega_t)^i$.

$$\begin{aligned}
\operatorname{argmax}_d V(d) &= \operatorname{argmax}_d (H(d) - H(\vec{f}) - \log Z + \sum_t \sum_{i=0}^{k_t} id(t, i) \log \omega_t) \\
&= \operatorname{argmax}_d (H(d) + \sum_t \sum_{i=0}^{k_t} id(t, i) \log \omega_t) \\
&= \operatorname{argmax}_d \sum_t \sum_{i=0}^{k_t} [-d(t, i) \log d(t, i) + id(t, i) \log \omega_t] \\
&= \operatorname{argmax}_d \sum_t \sum_{i=0}^{k_t} [-d(t, i) \log d(t, i) + d(t, i) \log(q(t, i)Y)] \\
&= \operatorname{argmax}_d \sum_t \sum_{i=0}^{k_t} [-d(t, i) \log d(t, i) + d(t, i) \log q(t, i) + d(t, i) \log Y] \\
&= \operatorname{argmax}_d \log Y + \sum_t \sum_{i=0}^{k_t} [-d(t, i) \log d(t, i) + d(t, i) \log q(t, i)] \\
&= \operatorname{argmin}_d \sum_t \sum_{i=0}^{k_t} [d(t, i) \log d(t, i) - d(t, i) \log q(t, i)] \\
&= \operatorname{argmin}_d \sum_t \sum_{i=0}^{k_t} d(t, i) \log \frac{d(t, i)}{q(t, i)} \\
&= \operatorname{argmin}_d H(d||q).
\end{aligned}$$

By definition, d_q^* minimizes $H(d||q)$. It is unique because H (and thus V) is a strictly concave function on a closed convex set. \square

Lemma A.1.4 tells us that the most likely distributions of money to be observed are those with low relative entropy to q . Among all distributions in $\Delta_{\vec{f}, m, k'}$, relative entropy is minimized by d_q^* . However, given n , it is quite possible that d_q^* is not $d^{\vec{x}}$ for any \vec{x} . For example, if $d_q^*(t, i) = 1/3$ for some t and i , but $f_i h n = 16$, then $d^{\vec{x}}(t, i) = d_q^*(t, i)$ only if exactly $16/3$ agents of type t to have i dollars, which cannot be the case. However, as the following lemma shows, for sufficiently

large n , we can always find a $d^{\vec{x}}$ that is arbitrarily close to d_q^* . For convenience, we use the 1-norm as our notion of distance.

Lemma A.1.5. *For all ϵ , there exists n_ϵ such that, if $n > n_\epsilon$, then for some state \vec{x} , $\|d^{\vec{x}} - d_q^*\| < \epsilon$.*

Proof. Given n , we construct $d \in \Delta_{\vec{f}, m, k}$ that is of the form $d^{\vec{x}}$ and is close to d_q^* in a number of steps. As a first step, for all t and i , let $d_1(t, i)$ be the result of rounding $d_q^*(t, i)$ to the nearest $1/hn$ (where ties are broken arbitrarily). The function d_1 may not be in $\Delta_{\vec{f}, m, k}$; we make minor adjustments to it to get a function in $\Delta_{\vec{f}, m, k}$. First, note that we may not have as is $d_1(t, i)$ for all t and i , we $\sum_i d_1(t, i) \neq f_t$. Since f_t is a multiple of $1/hn$, can get a function d_2 that satisfies these constraints by modifying each term $d_1(t, i)$ by either adding $1/hn$ to it, subtracting $1/hn$ from it, or leaving it alone. Such a function d_2 may still violate the final constraint that $\sum_{t,i} id_2(t, i) = m$. We construct a function d_3 that satisfies this constraint (while continuing to satisfy the constraint that $\sum_i d_3(t, i) = f_t$) as follows. Note that if we increase $d_2(t, i)$ by $1/hn$ and decrease $d_2(t, j)$ by $1/hn$, then we keep the keep $\sum_i d_2(t, i) = f_t$, and change $\sum_i id_2(t, i)$ by $(i - j)/hn$. Since each term $d_2(t, i)$ is a multiple of $1/hn$ and m is a multiple of $1/h$, we can perform these adjustments until all the constraints are satisfied.

The rounding to create d_1 changed each $d_1(t, i)$ by at most $1/hn$, so $\|d_q^* - d_1\|_1 \leq (\sum_t k_t + 1)/hn$. Since, each term $d_1(t, i)$ was changed by at most $1/hn$ to obtain $d_2(t, i)$, we have $\|d_1 - d_2\|_1 \leq (\sum_t k_t + 1)/hn$. Let $c = \max_t(\max(k_t - m, m))$. Each movement of up to $1/hn$ in the creation of d_1 and d_2 altered m by at most c/hn . Thus at most $2c$ movements are needed in the creation of d_3 for each pair (t, i) . Therefore, $\|d_2 - d_3\|_1 \leq (\sum_t k_t + 1)2c/hn$. By the triangle inequality, $\|d_q^* - d_3\|_1 \leq (\sum_t k_t + 1)(2c + 2)/hn$, which is $O(1/n)$. Hence, for n_ϵ sufficiently large, the resulting

d_3 will always be within distance ϵ of d_q^* .

Finally, we need to show that $d_3 = d^{\vec{x}}$ for some \vec{x} . Each $d_3(t, i)$ is a multiple of $1/hn$. There are hn agents in total, so we can find such an \vec{x} by taking any allocation of money such that $d_3(t, i)hn$ agents of type t have i dollars. \square

We are now ready to prove Theorem 2.2.1. We repeat the statement here for the reader's convenience.

THEOREM 2.2.1. *For all games $(T, \vec{f}, h, m, 1)$, all vectors \vec{k} of thresholds, and all $\epsilon > 0$, there exist $q \in \Delta_{\vec{f}, m, \vec{k}}$ and n_ϵ such that, for all $n > n_\epsilon$, there exists a round r^* such that, for all $r > r^*$, we have $\Pr(I_{q, n, \epsilon}^r = 1) > 1 - \epsilon$.*

Proof. From Lemma A.1.2, we know that, after a sufficient amount of time, the probability of being in state \vec{x} will be close to $\pi_{\vec{x}} = w_{\vec{x}}/Z$. Since \mathcal{M} converges to a limit distribution, it is sufficient to show that the theorem holds in the limit as $r \rightarrow \infty$. If the theorem holds in the limit for some $\epsilon' < \epsilon$, then we can take r large enough that the L1 distance between the distribution of the chain at time r and the limit distribution (i.e. treating the distributions as vectors and computing the sum of the absolute values of their differences) is at most $\epsilon - \epsilon'$.

The remainder of the proof is essentially that of Theorem 3.13 in [33] (applied in a very different setting). Let $V(d) = H(d) - H(\vec{f}) - \log Z + \sum_t \sum_{i=0}^{k_t} id(t, i) \log \omega_t$. We show there exists a value v_L such that, for all states \vec{x} such that $d^{\vec{x}}$ is not within ϵ of d_q^* , we have $V(d^{\vec{x}}) \leq v_L$, and a value $v_H > v_L$ such that $v_H = V(d^{\vec{y}})$ for some point \vec{y} such that $d^{\vec{y}}$ is within distance ϵ of d_q^* . Lemma A.1.3 then shows that it is exponentially more likely that $d^{\vec{x}^*} = d^{\vec{y}}$ than *any* distribution d such that

$V(d) \leq v_L$. If $\vec{x} = \vec{y}$ then $I_{q,n,\varepsilon}^r = 1$, and if $I_{q,n,\varepsilon}^r = 0$ then $V(d^{\vec{x}}) \leq v_L$, so this suffices to establish the theorem.

By Lemma A.1.4, the unique maximum of V on $\Delta_{\vec{f},m,\vec{k}}$ occurs at d_q^* . The set $\{d \in \Delta_{\vec{f},m,\vec{k}} \mid \|d_q^* - d\|_2 \geq \varepsilon\}$ is closed. V is a continuous function, so it takes some maximum v_L on this set. Pick some v_H such that $v_L < v_H < V(d_q^*)$. By the continuity of V , there exists an ϵ such that if $\|d_q^* - d\|_1 < \epsilon$ then $V(d) \geq v_H$. By Lemma A.1.5, for sufficiently large n , there is always some \vec{x} such that $\|d_q^* - d^{\vec{x}}\|_1 < \epsilon$. Thus, for some $\vec{x} \in X_{\varepsilon,q}$, $V(d^{\vec{x}}) \geq v_H$.

$\Pr(I_{q,n,\varepsilon}^r = 1) \geq \Pr(\vec{x} \in \{\vec{y} \mid d^{\vec{y}} = d^{\vec{x}}\})$. By Lemma A.1.3, $\Pr(I_{q,n,\varepsilon}^r = 1)$ is at least $1/F(hn)e^{hnV(d^{\vec{x}})} \geq 1/F(hn)e^{hmv_H}$. Now consider a \vec{y} such that $I_{q,n,\varepsilon}^r(\vec{y}) = 0$. By Lemma A.1.3, the probability that $d^{\vec{x}} = d^{\vec{y}}$ is at most $G(hn)e^{hnV(d^{\vec{y}})} \leq G(hn)e^{hmv_L}$. There are at most $(hn + 1)^{\sum_i (k_i + 1)}$ such points, a number which is polynomial in hn . Thus, for $G'(hn) = G(hn)(hn + 1)^{\sum_i (k_i + 1)}$, the probability that $I_{q,n,\varepsilon}^r = 0$ is at most $G'(hn)e^{hmv_L}$. The ratio of these probabilities is at most

$$\frac{G'(hn)e^{hmv_L}}{\frac{1}{F(hn)}e^{hmv_H}} = \frac{G'(hn)F(hn)}{e^{hn(v_H - v_L)}}.$$

This is the ratio of a polynomial to an exponential, so the probability of seeing a distribution of distance greater than ε from d_q^* goes to zero as n goes to infinity. \square

A.2 Proofs from Section 2.3

In this appendix, we provide the omitted proofs from Section 2.3.

The proof of Theorem 2.3.1 relies on modeling the game from the perspective of a single agent. Consider a vector \vec{k} of thresholds and the corresponding

strategy profile $\vec{S}(\vec{k})$. Fix an agent i of type t . Assume that all the agents other than i continue play their part of $\vec{S}(\vec{k})$. What is i 's best response? Since the set of agents is large, i 's choice of strategy will have (essentially) no impact on the distribution of money. By Theorem 2.2.1, the distribution of money will almost always be close to a distribution d^* . Suppose, the distribution were exactly d^* . Since we know the exact distribution of money and the thresholds used by the other agents, we can calculate the number of each type of agent that wish to volunteer and thus the probabilities that our single agent will be able to earn or spend a dollar. Thus, by assuming the distribution of money is always exactly d^* , we can model the game from the perspective of agent i as a Markov Decision Process (MDP). We show in Lemma A.2.2 that this MDP has an optimal threshold policy. (Threshold policies are known as *monotone* policies in the more general setting where there are more than two actions.) We then prove that any optimal policy for the MDP is an ϵ -best reply to the strategies of the other agents in the actual game.

Taking notation from Puterman [62], we formally define the MDP $\mathcal{P}_{G, \vec{S}(\vec{k}), t} = (S, A, p(\cdot | s, a), r(s, a))$ that describes the game where all the agents other than i are playing $\vec{S}(\vec{k})_{-i}$ and i has type t .

- $S = \{0, \dots, mhn\}$ is the set of possible states for the MDP (i.e., the possible amounts of money compatible with the distribution d^*).
- $A = \{0, 1\}$ is the set of possible actions for the agent, where 0 denotes not volunteering and 1 denotes volunteering iff another agent who has at least one dollar makes a request.
- p_u is the probability of earning a dollar, assuming the agent volunteers (given that all other agents have fixed their thresholds according to \vec{k} and

the distribution of money is exactly d^* . Each agent of type t' who wishes to volunteer can do so with probability $\beta_{t'}$. Assuming exactly the expected number of agents are able to volunteer, $v_{t'} = \beta_{t'}(f_{t'} - d^*(t', k_{t'}))n$ agents of type t' volunteer. Note that we are disregarding the effect of i in computing the $v_{t'}$, since this will have a negligible effect for large n . Using the $v_{t'}$ s, we can express p_u as the product of two probabilities: that some agent other than i who has a dollar is chosen to make a request and that i is the agent chosen to satisfy it. Thus,

$$p_u = \left(\sum_{t'} \rho_{t'}(f_{t'} - d^*(t', 0)) \right) \left(\frac{\chi_t \beta_t}{\sum_{t'} \chi_{t'} v_{t'}} \right). \quad (\text{A.2})$$

- p_d is the probability of agent i having a request satisfied, given that agent i has a dollar. Given that all agents are playing a threshold strategy, if the total number n of agents is sufficiently large, then it is almost certainly the case that some agent will always be willing and able to volunteer. Thus, we can take p_d to be the probability that agent i will be chosen to make a request; that is,

$$p_d = \frac{\rho_t}{n} \quad (\text{A.3})$$

- $r(s, a)$ is the (immediate) expected reward for performing action a in state s . Thus, $r(s, 0) = \gamma_t p_d$ if $s > 0$; $r(0, 0) = 0$; $r(s, 1) = \gamma_t p_d - \alpha_t p_u$ if $s > 0$; and $r(0, 1) = -\alpha_t p_u$.
- $p(s' | s, a)$ is the probability of being in state s' after performing action a in state s ; $p(s' | s, a)$ is determined by p_u and p_d ; specifically, $p(s+1 | s, 1) = p_u$, $p(s-1 | s, a) = p_d$ if $s > 0$, and the remainder of the probability is on $p(s | s, a)$ (i.e., $p(s | s, a) = 1 - (p(s+1 | s, 1) + p(s-1 | s, a))$).
- $u^*(s)$ is the expected utility of being in state s if agent i uses the optimal policy for the MDP $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$.

- $u(s, a)$ is the expected utility for performing action a in state s , given that the optimal strategy is followed after this action;

$$u(s, a) = r(s, a) + \delta \sum_{s'=0}^{mhn} p(s' | s, a) u^*(s').$$

To prove Theorem 2.3.1, we need two preliminary lemmas about the MDP $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$.

Lemma A.2.1. *For the MDP $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$, $u^*(s+2) + u^*(s) \leq 2u^*(s+1)$.*

Proof. The MDP $\mathcal{P}_{G, \vec{s}(\vec{k}), t}$ has an optimal stationary policy [62, Theorem 6.2.10] (a policy where the chosen action depends only on the current state). Let π be such a policy. Consider the policy π' starting in state $s+1$ that “pretends” it actually started in state s and is following π . More precisely, if $s_0 = s+1$ and $s_j > 0$ for $j = 0, \dots, k$, define $\pi'(s_0, s_1, \dots, s_k) = \pi(s_k - 1)$; otherwise, if $j \leq k$ is the least index such that $s_j = 0$, define $\pi'(s_0, \dots, s_k) = \pi(s_k)$. Given a history (s_0, \dots, s_k) , j is the random variable whose value is the minimum i such that $s_i = 0$ or ∞ if no such value exists. The definition of π' from π creates a bijection between histories that start in state $s+1$ and histories that start in state s , such that if h' corresponds to h , the probability of history h' with policy π' is the same as the probability of h with policy π . Technically, making the mapping a bijection requires the introduction of a new state $0'$, which intuitively represents the state where the agent has zero dollars and missed an opportunity to have a request satisfied last round because of it. More formally, we let $p(0' | 0, a) = p_d$ and $p(s | 0', a) = p(s | 0, a)$. With this change, the probabilities of corresponding histories are the same because the probability of transitioning from a state to the one “immediately below” it (where $s-1$ is immediately below s , $0'$ is immediately below 0 , and $0'$ is immediately below itself) is always p_d , and the probability of transitioning from

a state to the one “immediately above” it (where $s + 1$ is immediately above s , and 1 is immediately above $0'$) is always p_u .¹

This argument shows that an agent starting with $s + 1$ dollars “pretending” to start with s will have the same expected reward each round as an agent who actually started with s dollars, except during the first round j in a history such that $s_j = 0$. Thus (treating j as a random variable), we have

$$u^*(s + 1) \geq u^*(s) + E[\delta^j \gamma_t].$$

Similarly, we can use π starting from state $s + 2$ to define a policy π'' starting from state $s + 1$, where i “pretends” he has one more dollar and is using π , up to the first round j' that he is chosen to make a request with π in a state where he has no money (in which case he can make the request with π started from $s + 2$, but cannot make it with π'' started from $s + 1$); from that point on, he uses π . For corresponding histories, the utilities of an agent starting with $s + 1$ dollars and following π'' and an agent starting with $s + 2$ dollars and following π will be the same, except during round j' the agent following π will have a request satisfied but the agent following π'' will not. Thus,

$$u^*(s + 1) \geq u^*(s + 2) - E[\delta^{j'} \gamma_t].$$

Since, if i uses π , he will run out of money sooner if he starts with s dollars than if he starts with $s + 2$ dollars,

$$E[\delta^j \gamma_t] > E[\delta^{j'} \gamma_t].$$

Thus, $u^*(s + 2) + u^*(s) \leq 2u^*(s + 1)$. □

¹Note that this means that $0'$ is immediately below 0 but 1 is immediately above $0'$. This is intended, because $0'$ intuitively represents the state where the agent has 0 dollars and had a request go unsatisfied due to a lack of money in the previous round, so if he then earns a dollar he will have 1 dollar regardless of whether or not his request of two rounds previous was satisfied.

Lemma A.2.2. $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ has an optimal threshold policy.

Proof. As shown by Puterman [62, Lemma 4.7.1], it suffices to prove that $u(s, a)$ is subadditive. That is, we need to prove that, for all states s ,

$$u(s+1, 1) + u(s, 0) \leq u(s+1, 0) + u(s, 1). \quad (\text{A.4})$$

We consider here only the case that $s > 0$ (the argument is essentially the same if $s = 0$). Because $s > 0$, $r(s+1, a) = r(s, a)$, so (A.4) is equivalent to

$$\begin{aligned} & p_u u^*(s+2) + p_d u^*(s) + (1 - p_u - p_d) u^*(s+1) + p_d u^*(s-1) + (1 - p_d) u^*(s) \\ & \leq p_d u^*(s) + (1 - p_d) u^*(s+1) + p_u u^*(s+1) + p_d u^*(s-1) + (1 - p_u - p_d) u^*(s). \end{aligned}$$

This simplifies to

$$u^*(s+2) + u^*(s) \leq 2u^*(s+1),$$

which follows from Lemma A.2.1. □

We can now prove Lemma 2.3.1 and Theorem 2.3.1.

LEMMA 2.3.1. Consider the games $G_n = (T, \vec{f}, h, m, n)$ (where T , \vec{f} , h , and m are fixed, but n may vary). There exists a k such that for all n , s_k is an optimal policy for $\mathcal{P}_{G_n, \vec{S}(\vec{k}), t}$. The threshold k is the maximum value of κ such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u, p_d)}] \gamma_t, \quad (2.4)$$

where $J(\kappa, p_u, p_d)$ is a random variable whose value is the first round in which an agent starting with κ dollars, using strategy s_κ , and with probabilities p_u and p_d of earning a dollar and of being chosen given that he volunteers, respectively, runs out of money.

Proof. Fix n . Suppose that an agent is choosing between a threshold of κ and a threshold of $\kappa+1$. These policies only differ when the agent has κ dollars: he will

volunteer with the latter but not with the former. If he volunteers when he has κ dollars and is chosen, he will pay a cost of α_t and he will have $\kappa + 1$ dollars. As in the proof of Lemma A.2.1, we can define a bijection on histories such that, in corresponding histories of equal probability, an agent who started with κ dollars and is using s_κ will always have one less dollar than an agent who started with $\kappa + 1$ dollars and is using $s_{\kappa+1}$, until the first round r in which the agent using $s_{\kappa+1}$ has zero dollars. This means that in round $r - 1$ the agent using $s_{\kappa+1}$ had a request satisfied but the agent using s_κ was unable to because he had no money. Thus, if the agent volunteers when he has κ dollars and pays a cost of α_t in the current round, the expected value of being able to spend that dollar in the future is $E[(1 - (1 - \delta_t)/n)^{J(\kappa+1, p_u, p_d)}] \gamma_t$. Since this expectation is strictly increasing in κ (an agent with more money takes longer to spend it all), the maximum κ such that Equation (2.4) holds is an optimal threshold policy.

Taking the maximum value of κ that satisfies Equation (2.4) ensures that, for the n we fixed, we chose the maximum optimal threshold. We now need to show that this maximum optimal threshold is independent of n , which we do by showing that the expecting utility of every threshold policy s_κ is independent of n . The expected utility of a policy depends on the initial amount of money, but since an agent's current amount of money is a random walk whose transition probabilities are determined by p_u and p_d , there is a well-defined limit probability

$$x_i^* = \lim_{r \rightarrow \infty} \Pr(\text{agent has } i \text{ dollars in round } r)$$

determined by the ratio p_u/p_d (this is because the limit distribution satisfies the detailed balance condition: $x_i^* p_u = x_{i+1}^* p_d$). This distribution has the property that if the agent starts with i dollars with probability x_i^* , then in every round the probability he has i dollars is x_i^* . Thus, in each round his expected utility is

$\gamma p_d(1 - x_0^*) - \alpha p_u(1 - x_k^*)$. We can factor out n to write $p_u = p'_u/n$ and $p_d = p'_d/n$ where p'_u and p'_d are independent of n . Note that $p_u/p_d = p'_u/p'_d$, so the x_i^* 's are independent of n . Thus, we can rewrite the agent's expected utility for each round as c/n , where $c = \gamma p'_d(1 - x_0^*) - \alpha p'_u(1 - x_k^*)$ is independent of n . Therefore, the expected utility of s_k is

$$\sum_{r=0}^{\infty} \left(1 - \frac{1 - \delta_t}{n}\right)^r \frac{c}{n} = \frac{c}{1 - \delta_t},$$

which is independent of n . □

THEOREM 2.3.1. *For all games $G = (T, \vec{f}, h, m, n)$, all vectors \vec{k} of thresholds, and all $\varepsilon > 0$, there exist n_ε^* and $\delta_{\varepsilon, n}^*$ such that for all $n > n_\varepsilon^*$, types $t \in T$, and $\delta_t > \delta_{\varepsilon, n}^*$, an optimal threshold policy for $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ is an ε -best reply to the strategy profile $\vec{S}(\vec{k})_{-i}$ for every agent i of type t .*

Proof. By Lemma A.2.2, $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ has an optimal threshold policy. However, this might not be a best reply for agent i in the actual game if the other agents are playing $\vec{S}(\vec{k})$. $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ assumes that the probabilities of earning or spending a dollar in a given round are always exactly p_u and p_d respectively. Theorem 2.2.1 guarantees only that, in the game, the corresponding probabilities are close to p_u and p_d with high probability after some amount of time that can depend on n . A strategy S for player i in G defines a policy π_S for $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$ in the obvious way; similarly, a policy for the MDP determines a strategy for player i in the game. The expected utility of π_S is close to $U_i(S, \vec{S}(\vec{k})_{-i})$, but is, in general, not equal to it, because, as we noted, p_u and p_d may differ from the corresponding probabilities in the game. They differ for three reasons: (1) they are close, but not identical; (2) they are only close with high probability, and (3) they are only close after some amount of time. As we now show, given ε , the difference in

the expected utility due to each reason can be bounded by $\varepsilon/6$, so the expected utility of any strategy is within $\varepsilon/2$ of the value the corresponding policy in $\mathcal{P}_{G, \vec{S}(\vec{k}), t}$. Thus, an optimal strategy for the MDP is an ε -best reply.

As we have seen, the probabilities p_u and p_d are determined by the number of agents of each type that volunteer (i.e., the expressions $v_{t'}$ for each type t'). The distance between $d^{\vec{x}}$ and d^* bounds how much the actual number of agents of type t' that wish to volunteer in round r can differ from $v_{t'}/\beta_{t'}$. Even if exactly $v_{t'}/\beta_{t'}$ agents wish to volunteer for each type t , there might not be exactly $v_{t'}$ agents who actually volunteer because of the stochastic decision by nature about who can volunteer and because i cannot satisfy his own requests. However, for sufficiently large n , the effect on p_u and p_d from these two factors is arbitrarily close to zero. Applying Theorem 2.2.1, there exist n_1 and r_1 such that if there are at least n_1 agents, for all round $r > r_1$, $d^{\vec{x}}$ and d^* are sufficiently close that the difference between the utility of policy $\pi_{S'}$ in the MDP and $U_i((S', \vec{S}_{-i})$ in rounds $r > r_1$ where d^* is sufficiently close is at most $\varepsilon/6$.

Note that the maximum possible difference in utility between a round of the MDP and a round of the game is $\gamma + \alpha$ (if agent i spends a dollar rather than earning one). Applying Theorem 2.2.1 again, for $e = \varepsilon/6(\gamma + \alpha)$, there exist n_2 and r_2 such that the probability of the distribution not being within e of d^* is less than e . Thus, the difference between the expected utility of policy $\pi_{S'}$ in the MDP and $U_i((S', \vec{S}_{-i})$ in rounds $r > r_2$ where d^* is not sufficiently close is at most $e(\gamma + \alpha) = \varepsilon/6$.

Let $n_\varepsilon^* = \max(n_1, n_2)$ and $r^* = \max(r_1, r_2)$. The values of n_ε^* and r^* do not depend on δ , so we can take $\delta_{\varepsilon, n}^*$ to be sufficiently close to 1 that the total utility from the first r^* rounds is at most $\varepsilon/6$, completing the proof of the theorem. \square

Recall that BR_G maps a vector \vec{k} describing the threshold strategy for each type to a vector \vec{k}' of best replies.

LEMMA 2.3.2. *Consider the family of games $G_m = (T, \vec{f}, h, m, n)$ and the strategies $\vec{S}(\vec{k})$, for $mhn < \sum_t f_t k_t hn$. For this family of game, $\lambda_{m,\vec{k}}$ is non-decreasing in m and non-increasing in \vec{k} ; $p_u^{m,\vec{k}}$ is non-decreasing in m and non-increasing in \vec{k} ; and the function BR_G is non-decreasing in \vec{k} and non-increasing in m .*

Proof. We first show that that $\lambda_{m,\vec{k}}$ is monotone in m and \vec{k} . We then show that $p_u^{m,\vec{k}}$ is a monotone function of $\lambda_{m,\vec{k}}$ and that BR_G is a monotone function of $p_u^{m,\vec{k}}$, completing the proof.

We now show that $\lambda_{m,\vec{k}}$ is non-decreasing in m . Fix a vector of thresholds \vec{k} and let

$$g_{\vec{k}}(\lambda) = \sum_{t,i} i \frac{f_t \lambda^i q_{\vec{k}}(t, i)}{\sum_{j=0}^{k_t} \lambda^j q_{\vec{k}}(t, j)}, \quad (\text{A.5})$$

where $q_{\vec{k}}$ is the value of q from Equation (A.1) (we add the subscript \vec{k} to stress the dependence on \vec{k}). The definition of $\lambda_{m,\vec{k}}$ in Equations (2.2) and (2.3) in Lemma 2.2.1 ensures that, for all m , $m = g_{\vec{k}}(\lambda_{m,\vec{k}})$. A relatively straightforward computation shows that $g'_{\vec{k}}(\lambda) > 0$ for all λ . Thus, if $m' > m$, $g_{\vec{k}}(\lambda) = m$, and $g_{\vec{k}}(\lambda') = m'$, we must have $\lambda' > \lambda$. It follows that $\lambda_{m,\vec{k}}$ is increasing in m . (Note that $\lambda_{m,\vec{k}}$ is undefined for $m \geq \sum_t f_t k_t$, which is why monotonicity holds only for values of m such that $mhn < \sum_t f_t k_t$.)

We next show that $\lambda_{m,\vec{k}}$ is non-increasing in \vec{k} . Since we have a finite set of types, it suffices to consider the case where a single type t^* increases its threshold by 1. Let \vec{k} denote the initial vector of thresholds, and let \vec{k}' denote the vector of thresholds after agents of type t^* increase their threshold by 1; that is, $k_t = k'_t$ for $t \neq t^*$, and $k'_{t^*} = k_{t^*} + 1$.

The first step in showing that $\lambda_{m,\vec{k}}$ is non-increasing in \vec{k} is to show that $g_{\vec{k}'}(\lambda_{m,\vec{k}}) > g_{\vec{k}}(\lambda_{m,\vec{k}}) = m$. We do this by breaking the sum in the definition of g in Equation (A.5) into two pieces; those terms were $t \neq t^*$, and those where $t = t^*$.

It follows immediately from Equation (A.1) that there exists a constant c such that, for all i and $t \neq t^*$, we have $q_{\vec{k}'}(t, i) = cq_{\vec{k}}(t, i)$. It follows from Equation (2.2) that for all i and $t \neq t^*$, since $k_t = k'_t$, we have

$$i \frac{f_t \lambda_{m,\vec{k}}^i q_{\vec{k}'}(t, i)}{\sum_{j=0}^{k'_t} \lambda_{m,\vec{k}}^j q_{\vec{k}'}(t, j)} = i \frac{f_t \lambda_{m,\vec{k}}^i c q_{\vec{k}}(t, i)}{\sum_{j=0}^{k'_t} \lambda_{m,\vec{k}}^j c q_{\vec{k}}(t, j)} = i \frac{f_t \lambda_{m,\vec{k}}^i q_{\vec{k}}(t, i)}{\sum_{j=0}^{k_t} \lambda_{m,\vec{k}}^j q_{\vec{k}}(t, j)}; \quad (\text{A.6})$$

that is, the corresponding terms in the sum for $g_{\vec{k}'}(\lambda_{m,\vec{k}})$ and $g_{\vec{k}}(\lambda_{m,\vec{k}})$ are the same if $t \neq t^*$.

Now consider the corresponding terms for type t^* . First observe that for all $i < k'_t$,

$$\frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}'}(t^*, i)}{\sum_{j=0}^{k'_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}'}(t^*, j)} < \frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}}(t^*, i)}{\sum_{j=0}^{k_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}}(t^*, j)}; \quad (\text{A.7})$$

the two terms have essentially the same numerator (the use of $q_{\vec{k}'}$ instead of $q_{\vec{k}}$ cancels out as in Equation (A.6)), but the first has a larger denominator because $k'_{t^*} = k_{t^*} + 1$, so there is one more term in the sum. Since $f_{t^*} = \sum_{i=0}^{k_{t^*}} d_{q_{\vec{k}}}^*(t^*, i) = \sum_{i=0}^{k'_{t^*}} d_{q_{\vec{k}'}}^*(t^*, i)$, by Equations (2.2) and (2.3),

$$\sum_{i=0}^{k_{t^*}} \frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}}(t^*, i)}{\sum_{j=0}^{k_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}}(t^*, j)} = \sum_{i=0}^{k'_{t^*}} \frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}'}(t^*, i)}{\sum_{j=0}^{k'_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}'}(t^*, j)}. \quad (\text{A.8})$$

It follows that

$$\sum_{i=0}^{k'_{t^*}} i \frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}'}(t, i)}{\sum_{j=0}^{k'_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}'}(t, j)} > \sum_{i=0}^{k_{t^*}} i \frac{f_{t^*} \lambda_{m,\vec{k}}^i q_{\vec{k}}(t, i)}{\sum_{j=0}^{k_{t^*}} \lambda_{m,\vec{k}}^j q_{\vec{k}}(t, j)}. \quad (\text{A.9})$$

To see this, note that the two expressions above have the form $\sum_{i=0}^{k_{t^*}+1} i c_i$ and $\sum_{i=0}^{k_{t^*}} i d_i$, respectively. By Equation (A.8), $\sum_{i=0}^{k_{t^*}+1} c_i = \sum_{i=0}^{k_{t^*}} d_i = f_{t^*}$; by Equations

tion (A.7), $c_i < d_i$ for $i = 0, \dots, k_{r^*}$. Thus, in going from the right side to the left side, weight is being transferred from lower terms to $k_{r^*} + 1$.

Combining Equations (A.6) and (A.9) gives us $g_{\vec{k}'}(\lambda_{m,\vec{k}}) > g_{\vec{k}}(\lambda_{m,\vec{k}}) = m$, as desired. Since $g_{\vec{k}'}(\lambda_{m,\vec{k}}) = m$, by definition, it follows that $g_{\vec{k}'}(\lambda_{m,\vec{k}}) > g_{\vec{k}'}(\lambda_{m,\vec{k}'})$. Since, as shown above, $g_{\vec{k}'}$ is an increasing function, it follows that $\lambda_{m,\vec{k}} > \lambda_{m,\vec{k}'}$. Thus, $\lambda_{m,\vec{k}}$ is decreasing in \vec{k} .

We now show that the monotonicity of $\lambda_{m,\vec{k}}$ implies the monotonicity of $p_u^{m,\vec{k}}$. To do this, we show that, for all types t , $p_u^{m,\vec{k}} = p_d \lambda_{m,\vec{k}} \omega_t$. Since ω_t and p_d are independent of m and \vec{k} , it then follows that the monotonicity of $\lambda_{m,\vec{k}}$ implies the monotonicity of $p_u^{m,\vec{k}}$. (Recall that $\omega_t = \beta_t \chi_t / \rho_t$ was defined in Section 2.1.)

Fix a type t' . Then, dropping superscripts and subscripts on p_u , d , and λ for brevity, we have the following sequence of equalities (where the explanation for some of these lines is given following the equations):

$$p_u = \left(\sum_t \rho_t (f_t - d(t, 0)) \right) \left(\frac{\chi_{t'} \beta_{t'}}{n \sum_t \chi_t \beta_t (f_t - d(t, k_t))} \right) \quad (\text{A.10})$$

$$= \left(\frac{\sum_t \sum_{i=1}^{k_t} \rho_t d(t, i)}{\sum_t \sum_{i=0}^{k_t-1} \chi_t \beta_t d(t, i)} \right) \left(\frac{\chi_{t'} \beta_{t'}}{n} \right) \quad (\text{A.11})$$

$$= \left(\frac{\sum_t \sum_{i=0}^{k_t-1} \rho_t \lambda \omega_t d(t, i)}{\sum_t \sum_{i=0}^{k_t-1} \chi_t \beta_t d(t, i)} \right) \left(\frac{\chi_{t'} \beta_{t'}}{n} \right) \quad (\text{A.12})$$

$$= \lambda \left(\frac{\sum_t \sum_{i=0}^{k_t-1} \chi_t \beta_t d(t, i)}{\sum_t \sum_{i=0}^{k_t-1} \chi_t \beta_t d(t, i)} \right) (\omega_{t'} p_d) \quad (\text{A.13})$$

$$= \lambda \omega_{t'} p_d$$

Equation (A.10) is just the definition of p_u from Equation (A.2). Equation (A.11) follows from the observation that, by Equation (2.2), $f_t = \sum_i d(t, i)$. Equation (A.12) follows from the observation that, again by Equation (2.2), $d(t, i) = \omega_t \lambda d(t, i-1)$. Equation (A.13) follows from the definitions of ω_t and p_d (see Equa-

tion (A.3)). Thus, as required, $p_u^{m,\vec{k}} = p_d \lambda_{m,\vec{k}} \omega_t$.

Finally, we show that the monotonicity of $p_u^{m,\vec{k}}$ implies the monotonicity of BR_G . Let $\vec{k}'' = BR_G(\vec{k})$. By Lemma 2.3.1, k_t'' is the maximum value of κ such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u^{m,\vec{k}}, p_d)}] \gamma_t.$$

We (implicitly) defined the random variable $J(\kappa, p_u, p_d)$ as a function on histories. Instead, we can define $J(\kappa, p_u, p_d)$ as a function on random bitstrings (which intuitively determine a history). With this redefinition, it is clear that, if $p_u < p'_u$, for all bitstrings b , we have $J(\kappa, p_u, p_d)(b) < J(\kappa, p'_u, p_d)(b)$. It easily follows that

$$E[(1 - (1 - \delta_t)^{J(\kappa, p'_u, p_d)}] < E[(1 - (1 - \delta_t)^{J(\kappa, p_u, p_d)}]$$

for all κ . Thus, the monotonicity of BR_G follows from the monotonicity of $p_u^{m,\vec{k}}$. \square

LEMMA 2.3.3. *For all games $G = (T, \vec{f}, h, m, n)$, there exists a $\delta^* < 1$ such that if $\delta_t > \delta^*$ for all t , there is a vector \vec{k} of thresholds such that $BR_G(\vec{k}) > \vec{k}$.*

Proof. Take \vec{k} to be such that $k_t = \lceil m \rceil + 1$ for each type t . Then by Theorem 2.3.1, there exists a \vec{k}' such that $BR_G(\vec{k}) = \vec{k}'$. By Lemma 2.3.1, k_t' is the maximum value of κ such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u^{\vec{k}}, p_d)}] \gamma_t. \quad (2.4)$$

As δ_t approaches 1, $E[(1 - (1 - \delta_t)/n)^{J(\kappa, p_u^{\vec{k}}, p_d)}]$ approaches 1, and so the right hand side of Equation (2.4) approaches γ_t . For any standard agent, $\alpha_t < \gamma_t$. Thus, there exists a δ_t such that

$$\alpha_t \leq E[(1 - (1 - \delta_t)/n)^{J(k_t, p_u^{\vec{k}}, p_d)}] \gamma_t.$$

For this choice of δ_t , we must have $k_t' \geq k_t + 1 > k_t$. Take $\delta^* = \max_t \delta_t$. \square

A.3 Proofs from Section 3.2

THEOREM 3.2.3. *Fix a game G and vector of thresholds \vec{k} . Let $R_{\vec{k},t} = p_u^{\vec{k},t}/p_d^t$. In the limit as the number of rounds goes to infinity, the fraction of the agent's requests that have an agent willing and able to satisfy them that get satisfied is $(R_{\vec{k},t} - R_{\vec{k},t}^{k_t+1})/(1 - R_{\vec{k},t}^{k_t+1})$ if $R_{\vec{k},t} \neq 1$ and $k_t/(k_t + 1)$ if $R_{\vec{k},t} = 1$.*

Proof. Consider the Markov chain \mathcal{M} that results from fixing the agent's policy to s_{k_t} in $\mathcal{P}_{G,\vec{s}(\vec{k}),t}$. \mathcal{M} satisfies the requirements given in Theorem A.1.1 to have a limit distribution. It can be easily verified that the distribution gives the agent probability $R^i(1-R)/(1-R^{k+1})$ of having i dollars if $R \neq 1$ and probability $1/(k+1)$ if $R = 1$ satisfies the detailed balance condition and thus is the limit distribution. This gives the probabilities given in the theorem. \square

THEOREM 3.2.4. *Suppose that t and s are two types that agree except for the value of χ , and that $\chi_t < \chi_s$. If $\vec{k} = (k_t, k_s)$ is an ε -Nash equilibrium for $G = (\{t, s\}, \vec{f}, h, m, n)$ with social welfare w , then there exist m' , and n' such that $\vec{k}' = (k_s)$ is an ε -Nash equilibrium for $G'_{m',n'} = (\{t\}, \{1\}, h, m', n')$ with social welfare greater than w .*

Proof. We prove the theorem by finding m' , and n' such that agents in $G'_{m'}$ that play some strategy k get essentially the same utility that an agent with sybils would by playing that strategy in G . Since k_s was the optimal strategy for agents with sybils in G , it must be optimal in $G'_{m',n'}$ as well. Since agents with sybils have utility at least as great as those without, social welfare will be at least as large in $G'_{m',n'}$ as in G .

Since an agent can earn a dollar only if he is able to satisfy the current request, $0 < p_u^{m,\vec{k},s} < \beta_s$. The constraint that $hm'n'$, the total amount of money, is a

natural number means that m' must be a rational number. For the moment, we ignore that constraint and allow m' to take on any value in $[0, k'_t]$. From Equation (A.2), $p_u^{m', \vec{k}', t}$ is continuous in $d_{q_{\vec{k}'}}^*$, which, by Lemma 2.2.1, is continuous in $\lambda_{m', \vec{k}'}$ and thus m' . We use this continuity to show that we can find a value of m' such that $p_u^{m, \vec{k}, s} = p_u^{m', \vec{k}', t}$. By Equation (2.3), if $m' = 0$ then $d_{q_{\vec{k}'}, m}^*(t, 0) = 1$, and if $m' = k'_t$ then $d_{q_{\vec{k}'}, m}^*(t, k'_t) = 1$. Combining these with Equation (A.2) gives $p_u^{0, \vec{k}', t} = 0$ and $p_u^{m', \vec{k}', t} = \beta_t$. Thus, by the Intermediate Value Theorem, there exists an m' such that $p_u^{m, \vec{k}, s} = p_u^{m', \vec{k}', t}$. For this choice of m' , observe that by Lemma 2.3.1, $\mathcal{P}_{G, \vec{S}(\vec{k}), s}$ and $\mathcal{P}_{G_{m'}, \vec{S}(\vec{k}'), t}$ have the same optimal threshold policy.

If m' is rational, say $m' = a/b$, take $n' = bn$; then $hm'n'$ is an integer and, by the argument above, \vec{k}' is an equilibrium for $G_{m', n'}$. Since $p_u^{m', \vec{k}'} = p_u^{\vec{k}, s} > p_u^{\vec{k}, t}$, we must have $\zeta_{G_{m', n'}} < \zeta_G$. Thus, social welfare has increased. If m' is not rational, we instead use a rational value m'' sufficiently close to m' that \vec{k}' is still an equilibrium for $G_{m', n'}$ and $\zeta_{G_{m'', n''}} < \zeta_G$. \square

A.4 Proof of Lemma 3.3.2

LEMMA 3.3.2. *If d is a fully-supported distribution of money with finite support, there exist an infinite number of explanations of d .*

Proof. Fix λ . The distribution d and λ determine an explanation \vec{f} as follows. By Equation (3.3), we need \vec{f} to satisfy $d(j) \sum_{i=0}^{K_d} d_\lambda(i, j)$.

Recall that K_d is the maximum value for which $d(K_d) > 0$. Start by consider-

ing f_{K_d} . By the definition of d_λ , $d_\lambda(i, j) = 0$ if $j > i$. Thus, the constraint becomes

$$d(K_d) = d_\lambda(K_d, K_d) = f_{K_d} \lambda^{K_d} / \left(\sum_{l=0}^{K_d} \lambda^l \right).$$

Take f_{K_d} to be the unique value that satisfies this equation. Once we have defined f_{K_d} , again apply the constraint and take f_{K_d-1} to be the unique value that satisfies

$$d(K_d - 1) = d_\lambda(K_d, K_d - 1) + d_\lambda(K_d - 1, K_d - 1) = f_{K_d} \lambda^{K_d-1} / \left(\sum_{l=0}^{K_d} \lambda^l \right) + f_{K_d-1} \lambda^{K_d-1} / \left(\sum_{l=0}^{K_d-1} \lambda^l \right).$$

Iterating this process uniquely defines \vec{f} as the unique value that satisfies

$$d(j) = \sum_{i=j}^{K_d} d_\lambda(i, j) = \sum_{i=j}^{K_d} f_i \lambda^j / \left(\sum_{l=0}^i \lambda^l \right),$$

or

$$f_i = \left(\sum_{l=0}^i \lambda^l \right) / \lambda^i \left(d(i) - \sum_{j=i+1}^{K_d} f_j \lambda^i / \left(\sum_{l=0}^j \lambda^l \right) \right).$$

However, \vec{f} may not be an explanation, since some f_j may be negative. This happens exactly when

$$d(i) < \sum_{j=i+1}^{K_d} f_j \lambda^i / \left(\sum_{l=0}^j \lambda^l \right). \quad (\text{A.14})$$

As λ grows large, the right-hand side of (A.14) tends to 0. Since d is fully-supported, we must have $d(i) > 0$. Thus, we can ensure that (A.14) does not hold for any i by taking λ sufficiently large. Thus, for sufficiently large λ , \vec{f} provides an explanation for d . Continuing to increase λ will give an infinite number of different explanations. \square

BIBLIOGRAPHY

- [1] I. Abraham, D. Dolev, R. Gonen, and J.Y. Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 53–62, 2006.
- [2] E. Adar and B. A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.
- [3] K. G. Anagnostakis and M. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 524–533, 2004.
- [4] Christina Aperjis and Ramesh Johari. A peer-to-peer system as an exchange economy. In *GameNets '06: Proceeding from the 2006 Workshop on Game Theory for Communications and Networks*, page 10, 2006.
- [5] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.
- [6] R. J. Aumann. Acceptable points in general cooperative n -person games. In A.W. Tucker and R.D. Luce, editors, *Contributions to the Theory of Games IV, Annals of Mathematical Studies 40*, pages 287–324. Princeton University Press, Princeton, N. J., 1959.
- [7] Alvin AuYoung, Brent Chun, Chaki Ng, David Parkes, Amin Vahdat, and Alex Snoeren. Practical market-based resource allocation. Technical report, UC San Diego, 2007. CS2007-0901.
- [8] Mira Belenkiy, Melissa Chase, C. Christopher Erway, John Jannotti, Alptekin Küpçü, Anna Lysyanskaya, and Eric Rachlin. Making p2p accountable without losing privacy. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA, October 29, 2007*, pages 31–40, 2007.
- [9] Mattias Blonski. Equilibrium characterization in large anonymous games. Technical report, U. Mannheim, 2001.
- [10] Avrim Blum, Eyal Even-Dar, and Katrina Ligett. Routing without regret: on convergence to Nash equilibria of regret-minimizing algorithms in rout-

- ing games. In *Proc 25th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 45–52, 2006.
- [11] Avrim Blum and Yishay Mansour. Learning, regret minimization, and equilibria. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*, pages 79–102. Cambridge University Press, 2007.
 - [12] Michael H. Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *17th Int. Conf. on Machine Learning (ICML 2000)*, pages 89–94, 2000.
 - [13] Michael H. Bowling and Manuela M. Veloso. Rational and convergent learning in stochastic games. In *17th Int. Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 1021–1026, 2001.
 - [14] Richard T. Boylan. Laws of large numbers for dynamical systems with randomly matched individuals. *Journal of Economic Theory*, 57:473–504, 1992.
 - [15] J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. Parkes, M. Seltzer, J. Shank, and S. Youssef. Egg: An extensible and economics-inspired open grid computing platform. In *Third Workshop on Grid Economics and Business Models (GECON)*, pages 140–150, 2006.
 - [16] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning and Games*. Cambridge University Press, 2006.
 - [17] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Schneidman, A. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Second IEEE Workshop on Embedded Networked Sensors*, pages 19–28, 2005.
 - [18] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI-97 Workshop on Multiagent Learning*, pages 746–752, 1998.
 - [19] Bram Cohen. Incentives build robustness in BitTorrent. In *First Workshop on the Economics of Peer-to-Peer Systems (P2PECON)*, 2003.
 - [20] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.

- [21] Constantinos Daskalakis and Christos H. Papadimitriou. Computing equilibria in anonymous games. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, October 20-23, 2007, Providence, RI, USA, pages 83–93, 2007.
- [22] J. R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260, 2002.
- [23] G. Ellison. Cooperation in the prisoner’s dilemma with anonymous random matching. *Review of Economic Studies*, 61:567–588, 1994.
- [24] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM Conference on Electronic Commerce (EC)*, pages 102–111, 2004.
- [25] Dean P. Foster and Peyton Young. Regret testing: Learning to play Nash equilibrium without knowing you have an opponent. *Theoretical Economics*, 1:341–367, 2006.
- [26] E. J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.
- [27] Eric J. Friedman and Scott Shenker. Learning and implementation on the internet. 1998.
- [28] Drew Fudenberg and David Levine. *Theory of Learning in Games*. MIT Press, 1998.
- [29] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.
- [30] Fabrizio Germano and Gábor Lugosi. Global Nash convergence of Foster and Young’s regret testing. *Games and Economic Behavior*, 60(1):135–154, July 2007.
- [31] Ronen Gradwohl and Omer Reingold. Fault tolerance in large games. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC-2008)*, Chicago, IL, USA, June 8-12, 2008, pages 274–283, 2008.
- [32] Amy Greenwald, Eric J. Friedman, and Scott Shenker. Learning in networks contexts: Experimental results from simulations. *Games and Economic Behavior*, 35(1-2):80–123, 2001.

- [33] A. J. Grove, J. Y. Halpern, and D. Koller. Random worlds and maximum entropy. *J. Artif. Intell. Res. (JAIR)*, 2:33–88, 1994.
- [34] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Conference on the World Wide Web (WWW)*, pages 403–412, 2004.
- [35] M. Gupta, P. Judge, and M. H. Ammar. A reputation system for peer-to-peer networks. In *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 144–152, 2003.
- [36] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.
- [37] Sergiu Hart and Andreu Mas-Colell. A reinforcement learning procedure leading to correlated equilibrium. In Gerard Debreu, Wilhelm Neuefeind, and Walter Trockel, editors, *Economic Essays*, pages 181–200. Springer, 2001.
- [38] C. Hauert, A. Traulsen, H. Brandt, M. A. Nowak, and K. Sigmund. Via freedom to coercion: The emergence of costly punishment. *Science*, 316:1905–1907, 2007.
- [39] Ara Hayrapetyan, Éva Tardos, and Tom Wexler. The effect of collusion in congestion games. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 89–98, 2006.
- [40] T. Hens, K. R. Schenk-Hoppe, and B. Vogt. The great Capitol Hill baby sitting co-op: Anecdote or evidence for the optimum quantity of money? *J. of Money, Credit and Banking*, 9(6):1305–1333, 2007.
- [41] Ed Hopkins. Learning, matching, and aggregation. *Games and Economic Behavior*, 26:79–110, 1999.
- [42] Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.
- [43] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on Gnutella revisited: The bell tolls? *IEEE Distributed Systems Online*, 6(6), 2005.
- [44] J. Ioannidis, S. Ioannidis, A. D. Keromytis, and V. Prevelakis. Fileteller: Paying and getting paid for file storage. In *Financial Cryptography*, pages 282–299, 2002.

- [45] Ithaca Hours Inc. Ithaca hours, 2005. <http://www.ithacahours.org/>.
- [46] E. T. Jaynes. Where do we stand on maximum entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism*, pages 15–118. MIT Press, Cambridge, Mass., 1978.
- [47] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 4:237–285, 1996.
- [48] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Conference on the World Wide Web (WWW)*, pages 640–651, 2003.
- [49] M. Kandori. Social norms and community enforcement. *Review of Economic Studies*, 59:63–80, 1992.
- [50] Paul Krugman. *The Accidental Theorist*. Norton, 1999.
- [51] L. Lovasz and P. Winkler. Mixing of random walks and other diffusions on a graph. In *Surveys in Combinatorics, 1993, Walker (Ed.), London Mathematical Society Lecture Note Series 187, Cambridge University Press*. 1995.
- [52] Geoffrey Mainland, Laura Kang, Sébastien Lahaie, David C. Parkes, and Matt Welsh. Using virtual markets to program global behavior in sensor networks. In *Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22, 2004*, page 1, 2004.
- [53] J. R. Marden, G. Arslan, and J. S. Shamma. Connections between cooperative control and potential games. In *Proceedings of the 2007 European Control Conference (ECC)*, 2007.
- [54] J. R. Marden, H. P. Young, G. Arslan, and J. S. Shamma. Payoff-based dynamics for multi-player weakly acyclic games. *SIAM Journal on Control and Optimization*, 48(1):373–396, February 2009.
- [55] Jason R. Marden, Gürdal Arslan, and Jeff S. Shamma. Regret based dynamics: convergence in weakly acyclic games. In *6th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 42–49, 2007.
- [56] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, Oxford, U.K., 1995.

- [57] P. Milgrom and J. Roberts. Rationalizability, learning, and equilibrium in games with strategic complementarities. *Econometrica*, 58(6):1255–1277, 1990.
- [58] M. S. Miller and K. E. Drexler. Markets and computation: Agoric open systems. In B. Huberman, editor, *The Ecology of Computation*, pages 133–175. Elsevier, 1988.
- [59] C. Ng, P. Buonadonna, B. Chun, A. Snoeren, and A. Vahdat. Addressing strategic behavior in a deployed microeconomic resource allocator. In *Third Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, pages 99–104, 2005.
- [60] M. Osborne and A. Rubenstein. *A Course in Game Theory*. MIT Press, 1994.
- [61] Ryan S. Peterson and Emin Gün Sirer. Antfarm: Efficient content distribution with managed swarms. In *Proceedings of Networked Systems Design and Implementation (NSI), Boston, Massachusetts, April 2009*, 2009.
- [62] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.
- [63] D. M. Reeves, B. M. Soule, and T. Kasturi. Group decision making with Yootles. <http://ai.eecs.umich.edu/people/dreeves/yootles/yootles.pdf>.
- [64] S. I. Resnick. *Adventures in Stochastic Processes*. Birkhauser, 1992.
- [65] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.
- [66] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Stanford, 2003.
- [67] Michael Sirivianos, Jong Han Park, Xiaowei Yang, and Stanislaw Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *Proceedings of the 2007 USENIX Annual Technical Conference, June 17-22, 2007, Santa Clara, CA, USA*, pages 157–170, 2007.
- [68] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Stelin, and A. Yu. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1):48–63, 1996.

- [69] J. Sweeney and R. J. Sweeney. Monetary theory and the great capitol hill babysitting co-op crisis: Comment. *J. of Money, Credit and Banking*, 9(1):86–89, 1977.
- [70] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
- [71] Gerald Tesauro and Jeffrey O. Kephart. Pricing in agent economies using multi-agent Q-learning. *Autonomous Agents and Multi-Agent Systems*, 5(3):289–304, 2002.
- [72] R. H. Timberlake. Private production of scrip-money in the isolated community. *J. of Money, Credit and Banking*, 19(4):437–447, 1987.
- [73] D. M. Topkis. Equilibrium points in nonzero-sum n-person submodular games. *Siam Journal of Control and Optimization*, 17:773–787, 1979.
- [74] Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Journal of Autonomous Agents and Multi-agent Systems*, 14:239–269, 2007.
- [75] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: a secure economic framework for peer-to-peer resource sharing. In *First Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2003.
- [76] Kevin Walsh and Emin Gün Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Third Symp. on Network Systems Design & Implementation (NSDI)*, pages 1–14, 2006.
- [77] Washington Metropolitan Area Transit Commission. Scrip system of the D.C. transit system, 1970.
- [78] Christopher J.C.H. Watkins and Peter Dayan. Technical note Q-learning. *Machine Learning*, 8:279–292, 1992.
- [79] L. Xiong and L. Liu. Building trust in decentralized peer-to-peer electronic communities. In *International Conference on Electronic Commerce Research (ICECR)*, 2002.
- [80] Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. The effect of false-name bids in combinatorial auctions: new fraud in internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.

- [81] H. Peyton Young. Learning by trial and error. *Games and Economic Behavior*, 65(2):626–643, March 2009.
- [82] Shanyu Zhao, Virginia Mary Lo, and Chris GauthierDickey. Result verification and trust-based scheduling in peer-to-peer grids. In *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P 2005)*, 31 August - 2 September 2005, Konstanz, Germany, pages 31–38, 2005.